



Integrated Cloud Applications & Platform Services

Oracle Database: Administration Workshop

Student Guide

D105916GC10 | D105990

Learn more from Oracle University at education.oracle.com



Author

Donna Keesling

**Technical Contributors
and Reviewers**

Darryl Balaski
Trevor Bowen
Jody Glover
Joel Goodman
Jeff Ferreira
Mike Fitch
Hans Forbrich
Mark Fuller
Dominique Jeunot
Markiko Nakamura
Sailaja Pasupuleti
Padmaja Potineni
Bert Rich
Kathy Rich
James Spiller
Sarika Surampudi
Sravanti Tatiraju
Randy Urbano
Branislav Valny
Jean-Francois Verrier

Publisher

Jayanthy Keshavamurthy

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

1001282019

Contents

1 Introduction

- Objectives 1-2
- Course Objectives 1-3
- Introducing Oracle Database 1-4
- Oracle Database 18c 1-5
- Oracle Database 18c On-Premises Editions 1-6
- Oracle Database Standard Edition 2 1-8
- Oracle Database Options 1-9
- Oracle Management Packs 1-11
- What Is Oracle Cloud? 1-12
- Oracle Database Cloud Service: Overview 1-13
- Oracle Database Cloud Service Editions 1-14
- Oracle SQL and PL/SQL 1-15
- HR Schema 1-16
- Suggested Course Schedule 1-18
- Summary 1-19

2 Oracle Database Architecture

- Objectives 2-2
- Oracle Database Server Architecture: Overview 2-3
- Oracle Database Instance Configurations 2-4
- Connecting to the Database Instance 2-5
- Oracle Database Memory Structures 2-6
- Shared Pool 2-8
- Database Buffer Cache 2-10
- Redo Log Buffer 2-11
- Large Pool 2-12
- Java Pool 2-13
- Streams Pool 2-14
- Program Global Area (PGA) 2-15
- In-Memory Column Store: Introduction 2-16
- In-Memory Column Store: Overview 2-18
- Full Database In-Memory Caching 2-20
- Process Structures 2-21
- Database Writer Process (DBWn) 2-23

Log Writer Process (LGWR)	2-25
Checkpoint Process (CKPT)	2-27
System Monitor Process (SMON)	2-28
Process Monitor Process (PMON)	2-29
Recoverer Process (RECO)	2-30
Listener Registration Process (LREG)	2-31
Archiver Processes (ARCn)	2-32
Database Storage Architecture	2-33
Logical and Physical Database Structures	2-35
Segments, Extents, and Blocks	2-37
Tablespaces and Data Files	2-38
Default Tablespaces	2-39
SYSTEM and SYSAUX Tablespaces	2-40
Implementing Oracle Managed Files (OMF)	2-41
Oracle Container Database: Introduction	2-43
Multitenant Database	2-44
Multitenant Architecture	2-45
Default Tablespaces in the Multitenant Architecture	2-46
Application Containers	2-47
Automatic Storage Management	2-48
ASM Storage Components	2-49
Interacting with an Oracle Database: Memory, Processes, and Storage	2-50
Summary	2-52

3 Introduction to Oracle Database Cloud Service

Objectives	3-2
Oracle Cloud: Overview	3-3
Database Cloud Service Offerings	3-4
Infrastructure for Oracle Database Cloud Service	3-5
Database Cloud Service Architecture (OCI Classic)	3-6
Features and Tooling	3-7
Automated Database Provisioning	3-8
Additional Database Configuration Options	3-9
Summary	3-10
Practice 3: Overview	3-11

4 Creating DBCS Database Deployments

Objectives	4-2
Automated Database Provisioning	4-3
Creating a Database Deployment	4-4
How SSH Key Pairs Are Used	4-5

Creating an SSH Key Pair 4-6

Storage Used for Database Files 4-7

Summary 4-8

Practice 4: Overview 4-9

5 Accessing an Oracle Database

Objectives 5-2

Connecting to an Oracle Database Instance 5-3

Oracle Database Tools 5-5

Database Tool Choices 5-6

SQL*Plus 5-7

Oracle SQL Developer 5-9

Oracle SQL Developer: DBA Actions 5-10

SQL Developer Command Line (SQLcl) 5-11

Database Configuration Assistant (DBCA) 5-12

Oracle Enterprise Manager Database Express 5-13

Using the Database Home Page 5-15

Enterprise Manager Cloud Control 13c Features 5-16

Single Pane of Glass for Enterprise Management 5-18

Oracle Database Cloud Service Tools 5-19

Cloud Tooling 5-20

Accessing Tools and Features from the DBCS Console 5-21

Using Enterprise Manager Cloud Control 5-22

Summary 5-23

Practice 5: Overview 5-24

6 Managing DBCS Database Deployments

Objectives 6-2

Managing the Compute Node 6-3

Managing Network Access to DBCS (OCI Classic) 6-4

Enabling Access to a Compute Node Port (OCI Classic) 6-5

Scaling a Database Deployment 6-6

Patching DBCS 6-7

Using the DBCS Console to Manage Patches 6-8

Using the dbaascli Utility to Manage Patches 6-9

Summary 6-10

Practice 6: Overview 6-11

7 Managing Database Instances

Objectives 7-2

Working with Initialization Parameters 7-3

Initialization Parameters	7-5
Modifying Initialization Parameters	7-7
Viewing Initialization Parameters	7-9
Starting the Oracle Database Instance	7-11
Shutting Down an Oracle Database Instance	7-12
Comparing SHUTDOWN Modes	7-14
Opening and Closing PDBs	7-16
Working with the Automatic Diagnostic Repository	7-17
Automatic Diagnostic Repository	7-18
Viewing the Alert Log	7-19
Using Trace Files	7-20
Administering the DDL Log File	7-22
Querying Dynamic Performance Views	7-23
Considerations for Dynamic Performance Views	7-24
Data Dictionary: Overview	7-25
Querying the Oracle Data Dictionary	7-26
Summary	7-28
Practice 7: Overview	7-29

8 Oracle Net Services

Objectives	8-2
Oracle Net Services: Overview	8-3
Oracle Net Listener: Overview	8-4
The Default Listener	8-5
Establishing Oracle Network Connections	8-6
Connecting to an Oracle Database	8-7
Name Resolution	8-8
Establishing a Connection	8-9
User Sessions	8-10
Configuring Dynamic Service Registration	8-11
Configuring Static Service Registration	8-13
Naming Methods	8-15
Easy Connect	8-16
Local Naming	8-17
Directory Naming	8-18
Tools for Configuring and Managing Oracle Net Services	8-19
Defining Oracle Net Services Components	8-20
Advanced Connection Options	8-21
Testing Oracle Net Connectivity with tnsping	8-22

Configuring Communication Between Database Instances	8-23
Comparing Dedicated and Shared Server Configurations	8-25
Summary	8-27
Practice 8: Overview	8-28

9 Administering User Security

Objectives	9-2
Oracle Cloud User Roles and Privileges	9-3
Administering Oracle Cloud Users, Roles, and Privileges	9-4
Managing Oracle Cloud Compute Node Users	9-5
Database User Accounts	9-6
Oracle-Supplied Administrator Accounts	9-8
Creating Oracle Database Users in a Multitenant Environment	9-9
Schema-Only Account	9-10
Authenticating Users	9-11
Password Authentication	9-12
Password File Authentication	9-13
OS Authentication	9-14
OS Authentication for Privileged Users	9-16
Privileges	9-17
System Privileges	9-18
System Privileges for Administrators	9-20
Object Privileges	9-21
Granting Privileges in a Multitenant Environment	9-22
Granting and Revoking System Privileges	9-24
Granting and Revoking Object Privileges	9-25
Using Roles to Manage Privileges	9-26
Assigning Privileges to Roles and Assigning Roles to Users	9-27
Oracle-Supplied Roles	9-28
Creating and Granting Roles	9-29
Assigning Roles	9-31
Making Roles More Secure	9-32
Revoking Roles and Privileges	9-34
Profiles and Users	9-35
Creating Profiles in a Multitenant Architecture	9-36
Profile Parameters: Resources	9-38
Profile Parameters: Locking and Passwords	9-40
Oracle-Supplied Password Verification Functions	9-42
Assigning Profiles	9-44
Assigning Quotas	9-45
Applying the Principle of Least Privilege	9-47

Privilege Analysis 9-49
Privilege Analysis Flow 9-50
Summary 9-51
Practice 9: Overview 9-52

10 Creating PDBs

Objectives 10-2
Methods and Tools to Create PDBs 10-3
Creating PDBs from Seed 10-4
Creating a New PDB from PDB\$SEED 10-5
Examples: Creating a PDB from Seed 10-6
Cloning PDBs 10-7
Cloning Regular PDBs 10-8
Unplugging and Plugging in PDBs 10-9
Plugging an Unplugged Regular PDB into a CDB 10-10
Plugging Using an Archive File 10-11
Dropping PDBs 10-12
Summary 10-13
Practice 10: Overview 10-14

11 Creating Master Encryption Keys for PDBs

Objectives 11-2
Encryption in Database Cloud Service 11-3
Transparent Data Encryption (TDE): Overview 11-4
Components of TDE 11-5
Using TDE 11-6
Defining the Keystore Location 11-7
CDB and PDB Master Encryption Keys 11-8
Do You Need to Create and Activate a Master Encryption Key? 11-9
Creating and Activating a Master Encryption Key 11-10
Summary 11-11
Practice 11: Overview 11-12

12 Creating and Managing Tablespaces

Objectives 12-2
How Table Data Is Stored 12-3
Database Block Content 12-4
Creating Tablespace 12-5
Creating Permanent Tablespace in a CDB 12-8
Altering and Dropping Tablespace 12-9
Viewing Tablespace Information 12-11

Review: Implementing Oracle Managed Files (OMF)	12-12
Moving or Renaming Online Data Files	12-14
Examples: Moving and Renaming Online Data Files	12-15
Tablespace Encryption by Default in DBCS	12-16
Controlling Tablespace Encryption by Default	12-17
Managing the Software Keystore and Master Encryption Key	12-18
Creating an Encrypted Tablespace by Using a Nondefault Algorithm	12-19
Summary	12-20
Practice 12: Overview	12-21

13 Managing Storage Space

Objectives	13-2
Space Management Features	13-3
Block Space Management	13-4
Row Chaining and Migration	13-5
Free Space Management Within Segments	13-6
Types of Segments	13-7
Allocating Extents	13-8
Understanding Deferred Segment Creation	13-9
Controlling Deferred Segment Creation	13-10
Restrictions and Exceptions	13-11
Space-Saving Features	13-12
Private Temporary Tables	13-13
Table Compression: Overview	13-14
Compression for Direct-Path Insert Operations	13-15
Advanced Row Compression for DML Operations	13-16
Specifying Table Compression	13-17
Using Compression Advisor	13-18
Resolving Space Usage Issues	13-19
Monitoring Tablespace Space Usage	13-20
Reclaiming Space by Shrinking Segments	13-21
Shrinking Segments	13-22
Results of a Shrink Operation	13-23
Managing Resumable Space Allocation	13-24
Using Resumable Space Allocation	13-25
Resuming Suspended Statements	13-27
What Operations Are Resumable?	13-29
Summary	13-30
Practice 13: Overview	13-31

14 Managing Undo Data

Objectives 14-2
Undo Data: Overview 14-3
Transactions and Undo Data 14-5
Storing Undo Information 14-6
Comparing Undo Data and Redo Data 14-7
Managing Undo 14-8
Comparing SHARED Undo Mode and LOCAL Undo Mode 14-9
Configuring Undo Retention 14-10
Categories of Undo 14-11
Guaranteeing Undo Retention 14-12
Changing an Undo Tablespace to a Fixed Size 14-13
Temporary Undo: Overview 14-14
Temporary Undo Benefits 14-15
Enabling Temporary Undo 14-16
Monitoring Temporary Undo 14-17
Viewing Undo Information 14-18
Viewing Undo Activity 14-19
Summary 14-20
Practice 14: Overview 14-21

15 Moving Data

Objectives 15-2
Moving Data: General Architecture 15-3
Oracle Data Pump: Overview 15-4
Oracle Data Pump: Benefits 15-5
Data Pump Export and Import Clients 15-7
Data Pump Interfaces and Modes 15-8
Data Pump Import Transformations 15-10
SQL Loader: Overview 15-11
Comparing Loading Methods 15-13
Data Save Feature 15-14
Express Mode 15-15
External Tables 15-16
External Table Benefits 15-17
Migrating to Oracle Database Cloud Service: Considerations 15-18
Migrating to Oracle Database Cloud Service: Information Gathering 15-19
Applicable Migration Methods 15-20
Summary 15-22
Practice 15: Overview 15-23

16 Backup and Recovery Concepts

- Objectives 16-2
- DBA Responsibilities 16-3
- Categories of Failure 16-4
- Statement Failure 16-5
- User Process Failure 16-6
- Network Failure 16-7
- User Error 16-8
- Instance Failure 16-9
- Media Failure 16-10
- Understanding Instance Recovery 16-11
- The Checkpoint (CKPT) Process 16-12
- Redo Log Files and the Log Writer (LGWR) Process 16-13
- Automatic Instance Recovery or Crash Recovery 16-14
- Phases of Instance Recovery 16-15
- Tuning Instance Recovery 16-16
- Using the MTTR Advisor 16-17
- Comparing Complete and Incomplete Recovery 16-18
- The Complete Recovery Process 16-19
- The Point-in-Time Recovery Process 16-20
- Oracle Data Protection Solutions 16-22
- Flashback Technology 16-23
- Summary 16-24

17 Backup and Recovery Configuration

- Objectives 17-2
- Configuring for Recoverability 17-3
- Configuring the Fast Recovery Area 17-4
- Monitoring the Fast Recovery Area 17-5
- Multiplexing Control Files 17-6
- Redo Log Files 17-8
- Multiplexing the Redo Log 17-9
- Creating Archived Redo Log Files 17-10
- Archiver (ARCn) Process 17-11
- Archived Redo Log Files: Naming and Destinations 17-12
- Configuring ARCHIVELOG Mode 17-13
- Summary 17-14
- Practice 17: Overview 17-15

18 Creating Database Backups

- Objectives 18-2
- Understanding Types of Backups 18-3
- Backup Terminology 18-4
- Understanding Types of Backups 18-5
- RMAN Backup Types 18-6
- Using Recovery Manager (RMAN) 18-8
- Backing Up the Control File to a Trace File 18-9
- Using RMAN Commands to Create Backups 18-10
- Backing Up Databases on DBCS 18-11
- Backup Destination Choices 18-12
- Backup Configuration 18-13
- Creating an On-Demand Backup 18-14
- Customizing the Backup Configuration 18-15
- Summary 18-16
- Practice 18: Overview 18-17

19 Performing Database Recovery

- Objectives 19-2
- Opening a Database 19-3
- Keeping a Database Open 19-5
- Data Recovery Advisor 19-6
- Loss of a Control File 19-8
- Loss of a Redo Log File 19-9
- Loss of a Data File in NOARCHIVELOG Mode 19-11
- Loss of a Noncritical Data File in ARCHIVELOG Mode 19-12
- Loss of a System-Critical Data File in ARCHIVELOG Mode 19-13
- DBCS: Performing Recovery by Using the Console 19-14
- DBCS: Performing Recovery by Using the dbaascli Utility 19-15
- Summary 19-16
- Practice 19: Overview 19-17

20 Monitoring and Tuning Database Performance

- Objectives 20-2
- Performance Management Activities 20-3
- Performance Planning Considerations 20-4
- Database Maintenance 20-6
- Automatic Workload Repository (AWR) 20-7
- Automatic Database Diagnostic Monitor (ADDM) 20-8
- Advisory Framework 20-9
- Automated Maintenance Tasks 20-11

Server-Generated Alerts	20-12
Setting Metric Thresholds	20-13
Reacting to Alerts	20-14
Alert Types and Clearing Alerts	20-15
Database Server Statistics and Metrics	20-16
Performance Monitoring	20-17
Viewing Statistics Information	20-18
Monitoring Wait Events	20-20
Monitoring Sessions	20-21
Monitoring Services	20-22
Performance Tuning Methodology	20-23
Managing Memory Components	20-24
Automatic Memory Management	20-26
Automatic Shared Memory Management	20-28
Managing the SGA for PDBs	20-30
Managing the Program Global Area (PGA)	20-31
Managing the PGA for PDBs	20-33
Summary	20-34
Practice 20: Overview	20-35

21 Tuning SQL

Objectives	21-2
SQL Tuning Process	21-3
Oracle Optimizer	21-4
Optimizer Statistics	21-5
Optimizer Statistics Collection	21-6
Setting Optimizer Statistics Preferences	21-8
Optimizer Statistics Advisor	21-10
Optimizer Statistics Advisor Report	21-11
Executing Optimizer Statistics Advisor Tasks	21-12
SQL Plan Directives	21-13
Adaptive Execution Plans	21-14
SQL Tuning Advisor: Overview	21-16
SQL Access Advisor: Overview	21-18
SQL Performance Analyzer: Overview	21-19
Summary	21-21
Practice 21: Overview	21-22

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Introduction



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the differences in Oracle Database editions, options, and packs
- List the database offerings in Oracle Cloud
- Explain the sample database that will be used in the course practices



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Course Objectives

After completing this course, you should be able to:

- Describe Oracle Database architecture
- Explain Oracle Database Cloud Service (DBCS) architecture and features
- Create and manage DBCS database deployments
- Configure the database to support your applications
- Manage database security and implement auditing
- Implement basic backup and recovery procedures
- Move data between databases and files
- Employ basic monitoring procedures and manage performance



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Introducing Oracle Database

- Oracle provides cloud and on-premises offerings.
- The purpose of Oracle Database is to store, organize, and retrieve data for your applications.
- You can install Oracle Database in your environment (on-premises) or use Oracle Database in Oracle's environment (cloud).



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Database is a relational database management system (RDBMS). You can use Oracle Database in your own environment by installing the software. In the cloud, Oracle hosts Oracle Database and all its components (for example, networking and storage components) for you.

Oracle Database enables you to store, organize, and retrieve data for your applications. For example, a Human Resources department has a custom software application that uses Oracle Database on the back end to store its data about employees. The application is the front end and provides a user-friendly interface where users can update data in the database and retrieve data to create reports.

Oracle Database 18c

- First annual release of Oracle Database
- Will be released first on Oracle Cloud and Engineered Systems, with on-premises releases following
- Quarterly Release Updates (RUs) and Release Update Revisions (RURs) will be delivered



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Beginning with Oracle Database 18c, Oracle will deliver annual releases and quarterly updates of Oracle Database.

Oracle Database 18c On-Premises Editions

Oracle Database is available in the following editions, each suitable for different development and deployment scenarios:

- Oracle Database Personal Edition (PE)
- Oracle Database Standard Edition 2 (SE2)
- Oracle Database Enterprise Edition (EE)



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Database Personal Edition

Oracle Database Personal Edition supports single-user development and deployment environments that require full compatibility with Oracle Database Standard Edition One, Oracle Database Standard Edition, and Oracle Database Enterprise Edition.

Personal Edition includes all the components that are included with Enterprise Edition, as well as all the options that are available with Enterprise Edition. Personal Edition is available on Windows platforms only. The Management Packs are not included in Personal Edition.

Note: You cannot use the Oracle Real Application Clusters option with Personal Edition.

Oracle Database Standard Edition 2

Before SE2, there is SE (Standard Edition) and SE1 (Standard Edition1).

Oracle Database Standard Edition delivers the unprecedented ease of use, power, and performance of Standard Edition One, with support for larger machines and clustering of services with Oracle Real Application Clusters (Oracle RAC). Oracle RAC is not included in the Standard Edition of releases before Oracle Database 10g, nor is it an available option with those earlier releases.

Oracle Database Standard Edition One delivers unprecedented ease of use, power, and performance for workgroup, department-level, and web applications. From single-server environments for small business to highly distributed branch environments, Oracle Database Standard Edition One includes all the facilities necessary to build business-critical applications.

SE2 starts with Oracle Database 12c Release 1 (12.1.0.2).

SE2 supports Oracle RAC.

Oracle Database Enterprise Edition

Oracle Database Enterprise Edition provides the performance, availability, scalability, and security required for mission-critical applications such as high-volume online transaction processing (OLTP) applications, query-intensive data warehouses, and demanding Internet applications. Oracle Database Enterprise Edition contains all the components of Oracle Database and can be further enhanced with the purchase of the options and packs.

- Oracle Database Options
 - All the Oracle Database options can be purchased with Oracle Database Enterprise Edition.
 - You may not use the options, packs, or products without separately purchased licenses. The fact that these options, packs, or products may be included in product CDs, downloads, or described in documentation that you receive does not authorize you to use them without purchasing appropriate licenses.
- Oracle Management Packs
 - The management packs can be purchased only with Enterprise Edition.
 - The features in these packs are accessible through Oracle Enterprise Manager Database Control, Oracle Enterprise Manager Grid Control, and APIs provided with Oracle Database software.

Additional information about Oracle Database options and management packs is provided later in the lesson.

Oracle Database Standard Edition 2

- SE2 supports Oracle Real Application Clusters (RAC).
- SE2 supports single tenant but lacks the following features, options, and tools:
 - Parallel execution
 - Oracle Data Guard
 - Enterprise Manager Cloud Control
 - Management packs



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

SE2 supports the multitenant architecture, but with one pluggable database (PDB) only.

SE2 lacks the following features:

- Flashback Database (and all other Flashback features), tablespace point-in-time recovery, online table redefinition, Query Results Cache, Database Resource Manager
- Options such as In-Memory Database Cache, Oracle Partitioning, and Oracle Database Vault
- All management packs such as Oracle Tuning Pack

SE2 cannot be used with Oracle Enterprise Manager Cloud Control.

Oracle Database Options

Option	Description
Oracle Active Data Guard	Increases performance, availability, data protection, and return on investment wherever Data Guard is used for real-time data protection and availability
Oracle Advanced Analytics	Empowers data and business analysts to extract knowledge, discover new insights, and make predictions—working directly with large data volumes
Oracle Advanced Compression	Provides comprehensive data compression and Information Lifecycle Management (ILM) capabilities for all types of data
Oracle Advanced Security	Helps you protect sensitive information and comply with privacy and compliance regulations by enabling database encryption and data redaction
Oracle Database In-Memory	Enables any existing Oracle Database—compatible application to automatically and transparently take advantage of columnar in-memory processing, without additional programming or application changes
Oracle Database Vault	Enables you to control when, where, and by whom the database and application data can be accessed



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The table lists the separately licensed options that are available for use with Oracle Database. You must be licensed for an option in order to use any of its features.

See *Oracle Database Licensing Information User Manual* for details on each of the Oracle Database options, including the usability of options with each edition.

Oracle Database Options

Option	Description
Oracle Label Security	Provides sophisticated and flexible security based on row labels for fine-grained access control
Oracle Multitenant	Enables an Oracle database to function as a multitenant container database (CDB) that includes one or more pluggable databases (PDBs)
Oracle On-Line Analytical Processing (OLAP)	A full-featured OLAP server embedded in Oracle Database Enterprise Edition
Oracle Partitioning	Adds significant manageability, availability, and performance capabilities to large underlying database tables and indexes
Oracle Real Application Clusters (Oracle RAC)	A database computing environment that harnesses the processing power of multiple interconnected computers using clustering technology
Oracle Real Application Testing	Comprises a suite of features that help protect database applications from the undesirable impact of routine changes
Oracle Spatial and Graph	Includes advanced features for spatial data and analysis and for physical, network, and social graph applications



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Management Packs

Pack	Description
Oracle Cloud Management Pack for Oracle Database	Helps to set up a Database Cloud and operate the Database as a Service model
Oracle Data Masking and Subsetting Pack	Facilitates the creation of production-like data for nonproduction environments by replacing production data with fictitious yet realistic values
Oracle Database Lifecycle Management Pack for Oracle Database	Provides a comprehensive solution that helps database, system, and application administrators automate the processes required to manage the Oracle Database Lifecycle
Oracle Diagnostics Pack	Provides automatic performance diagnostic and advanced system monitoring functionality
Oracle Tuning Pack	Provides database administrators with expert performance management for the Oracle environment, including SQL tuning and storage optimizations



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

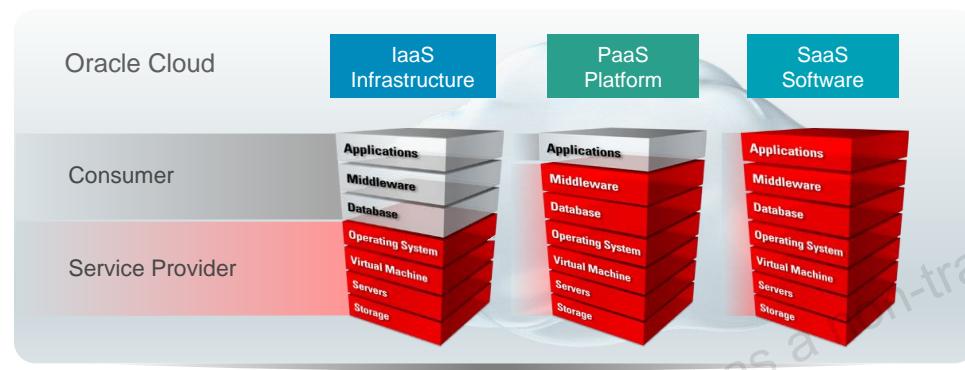
The table lists the separately licensed packs that are available for use with Oracle Database.

See *Oracle Database Licensing Information User Manual* for details on each of the management packs.

What Is Oracle Cloud?



Oracle Cloud is an enterprise cloud for business. Oracle Cloud offers self-service business applications delivered on an integrated development and deployment platform with tools to extend services and create new services rapidly.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

With **Software as a Service (SaaS)**, modern cloud applications from Oracle help you re-imagine your business. The best-of-breed SaaS applications in Oracle Cloud are integrated with social, mobile, and analytic capabilities to help you deliver the experiences customers expect, the talent to succeed, and the performance that the market demands.

Oracle Cloud Platform as a Service (PaaS) helps enterprise IT and independent software vendor (ISV) developers rapidly build and deploy rich applications or extend SaaS applications by using an enterprise-grade cloud platform based on the industry's leading database and application server.

Oracle Cloud Infrastructure as a Service (IaaS) is a comprehensive set of integrated, subscription-based infrastructure services that enable businesses to run any workload in an enterprise-grade cloud that is managed, hosted, and supported by Oracle.

For other services, refer to the Oracle Cloud website.

Oracle Database Cloud Service: Overview

- Oracle Database Cloud Service is a PaaS offering.
- With Oracle Database Cloud Service, you can:
 - Provision a full-featured dedicated Oracle database
 - Use cloud tooling to back up, patch, and manage the database
 - Avail of the complete administration privileges of the server and database to manage it as you need



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When you provision a database deployment in Oracle Database Cloud Service, you select the Oracle Database software release:

- Oracle Database 11g Release 2
- Oracle Database 12c Release 1
- Oracle Database 12c Release 2
- Oracle Database 18c Release 1

Note: Additional software releases may be added as they become available, and some releases may no longer be supported for new database deployment. Refer to “Oracle Database Software Release” in *Administering Oracle Database Cloud Service* for the current list of supported releases.

Oracle Database Cloud Service Editions



Edition	Included Options	Included Packs
Standard	None	None
Enterprise	None	None
Enterprise—High Performance	Advanced Analytics, Advanced Compression, Advanced Security, Database Vault, Label Security, Multitenant, OLAP, Partitioning, Real Application Testing, Spatial and Graph	Cloud Management for Oracle Database, Database Lifecycle Management, Data Masking and Subsetting, Diagnostics, Tuning
Enterprise—Extreme Performance	Active Data Guard, Advanced Analytics, Advanced Compression, Advanced Security, Database In-Memory, Database Vault, Label Security, Multitenant, OLAP, Partitioning, Real Application Clusters, Real Application Testing, Spatial and Graph	Cloud Management for Oracle Database, Database Lifecycle Management, Data Masking and Subsetting, Diagnostics, Tuning



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When you create a database deployment, you can choose from four software editions. The High Performance and Extreme Performance editions include the Oracle Database options and Oracle Management packs listed in the slide.

Oracle SQL and PL/SQL

- Oracle SQL is the language you use to perform operations on the data in an Oracle database. For example, selecting data from a database:


```
SQL> SELECT employee_id, first_name, last_name FROM employees
      WHERE employee_id=216 ORDER BY 1;
```

 - SELECT lists the database columns for which you want to view data.
 - FROM lists the tables that contain those database columns.
 - WHERE specifies column limits and table joins (this part essentially filters the rows of data).
 - ORDER BY specifies the columns by which the results are sorted.
- PL/SQL is a procedural extension to Oracle SQL.
 - It enables you to control the flow of a SQL program, use variables, and write error-handling procedures.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle SQL, DDL, and DML

Oracle SQL is the language you use to perform operations on the data in an Oracle Database. It's an implementation of the ANSI standard language called Structured Query Language (SQL) and provides features that extend beyond standard SQL. For example, you use SQL to create tables, query tables, and modify data in tables. A SQL statement can be thought of as a very simple, but powerful, computer program or instruction. Users specify the result that they want (for example, the names of employees), not how to derive it. A SQL statement is a string of SQL text such as the following:

```
SELECT first_name, last_name FROM employees;
```

Data definition language (DDL) includes statements that define or change a data structure (for example, CREATE TABLE and ALTER INDEX statements).

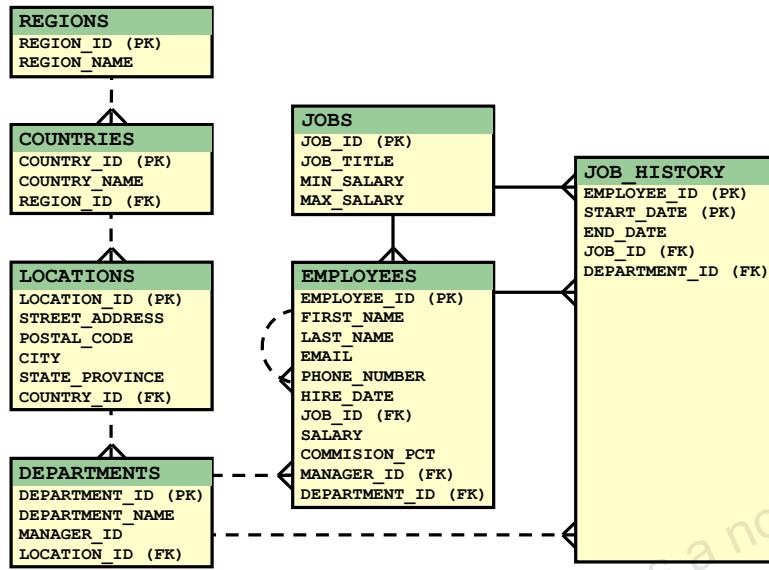
Data manipulation language (DML) includes statements such as SELECT, INSERT, UPDATE, and DELETE.

PL/SQL

PL/SQL is a procedural extension to Oracle SQL. PL/SQL is integrated with Oracle Database, enabling you to use all the Oracle Database SQL statements, functions, and data types. You can use PL/SQL to control the flow of a SQL program, use variables, and write error-handling procedures.

Oracle Database can also store program units written in Java. A Java stored procedure is a Java method published to SQL and stored in the database for general use. You can call existing PL/SQL programs from Java and Java programs from PL/SQL.

HR Schema



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Relational Database Models

Oracle Database uses a relational database model that organizes and presents the physical data as logical structures, such as tables (with columns and rows). These logical structures make the data understandable. A schema is a collection of logical structures that are owned by a database user and can include tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. In general, schemas include everything your application creates in the database.

HR Schema

The HR schema, which is shown in the slide and used throughout this course, groups seven related tables and was created by the `HR` user. A primary key (identified by `PK` in the slide) is a unique identifier for a row in a table. A foreign key (identified by `FK` in the slide) references a primary key. It uses the same value as the primary key. Together, primary keys and foreign keys are used to ensure data integrity. For example, in the `JOBS` table, each row uniquely identifies a job. Each job is given a job ID, and this ID is the primary key. Likewise, the `EMPLOYEES` table uniquely identifies each employee with an employee ID. The employee ID is also a primary key. The `EMPLOYEES` table contains a foreign key named `JOB_ID`, which references the `JOB_ID` primary key in the `JOBS` table. Solid lines in the diagram represent mandatory foreign key constraints and dashed lines represent optional foreign key constraints.

The following are some principal business rules implemented in the `HR` schema:

- Each department may be the employer of one or more employees. Each employee may be assigned to only one department.
- Each job must be a job for one or more employees. Each employee must be currently assigned to only one job.

- When an employee changes his or her department or job, a record in the `JOB_HISTORY` table records the start and end dates of the past assignments.
- The `JOB_HISTORY` table records are identified by a composite primary key: the `EMPLOYEE_ID` and the `START_DATE` columns.
- The `EMPLOYEES` table also has a foreign key constraint with itself. This is an implementation of the business rule: each employee may be reporting directly to only one manager. The foreign key is optional because the top employee does not report to another employee.

Suggested Course Schedule

Day	Lessons	Day	Lessons
1	<ul style="list-style-type: none">1. Introduction2. Oracle Database Architecture3. Introduction to Oracle Database Cloud Service4. Creating DBCS Database Deployments5. Accessing an Oracle Database	3	<ul style="list-style-type: none">10. Creating PDBs11. Creating Master Encryption Keys for PDBs12. Creating and Managing Tablespaces13. Managing Storage Space
2	<ul style="list-style-type: none">6. Managing DBCS Database Deployments7. Managing Database Instances8. Understanding Oracle Net Services9. Administering User Security	4	<ul style="list-style-type: none">14. Managing Undo Data15. Moving Data16. Backup and Recovery Concepts17. Backup and Recovery Configuration
		5	<ul style="list-style-type: none">18. Creating Database Backups19. Performing Database Recovery20. Monitoring and Tuning Database Performance21. SQL Tuning



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Describe the differences in Oracle Database editions, options, and packs
- List the database offerings in Oracle Cloud
- Explain the sample database that will be used in the course practices



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Oracle Database Architecture



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

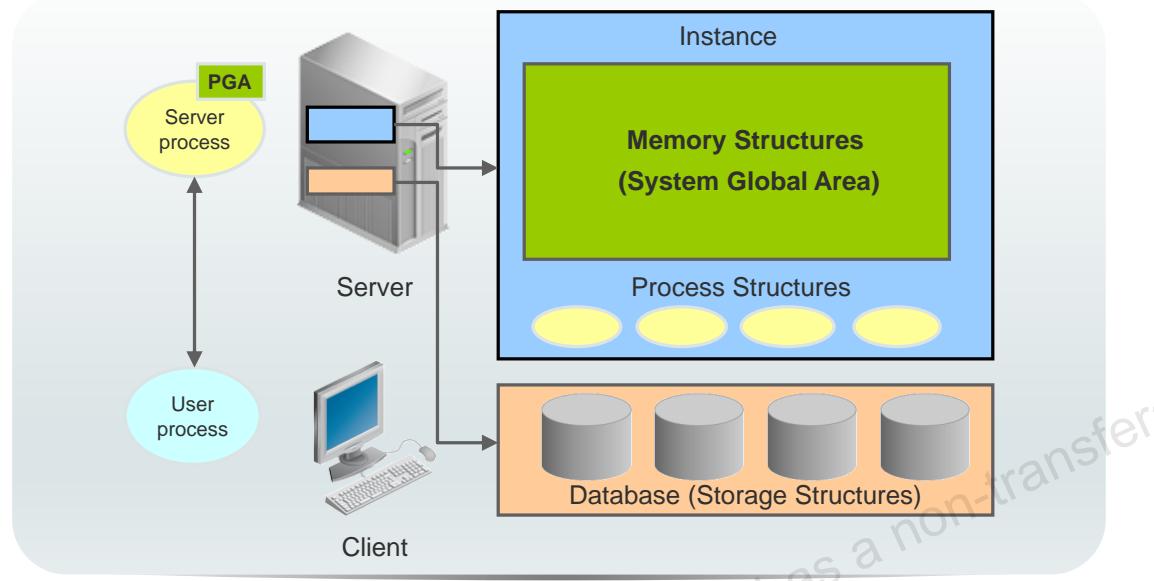
- List the major architectural components of Oracle Database
- Explain memory structures
- Describe background processes
- Correlate logical and physical storage structures
- Describe multitenant architecture



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Database Server Architecture: Overview



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

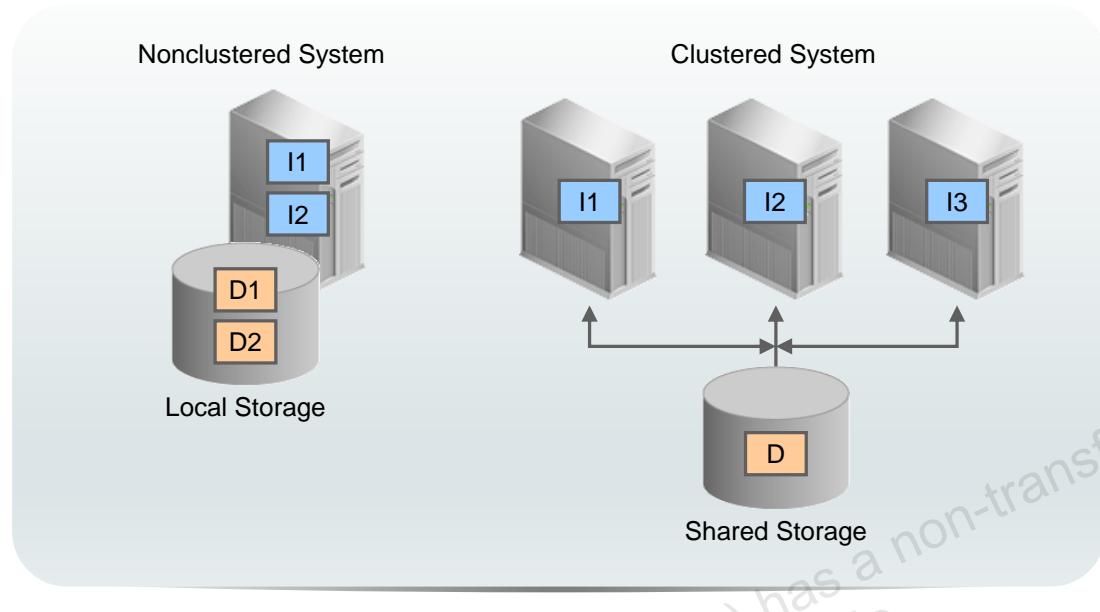
There are three major structures in Oracle Database server architecture: memory structures, process structures, and storage structures. A basic Oracle database system consists of an Oracle database and a database instance.

The database consists of both physical structures and logical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting access to logical storage structures.

The instance consists of memory structures and background processes associated with that instance. Every time an instance is started, a shared memory area called the System Global Area (SGA) is allocated, and the background processes are started. Processes are jobs that work in the memory of computers. A process is defined as a “thread of control” or a mechanism in an operating system that can run a series of steps. After starting a database instance, the Oracle software associates the instance with a specific database. This is called *mounting the database*. The database is then ready to be opened, which makes it accessible to authorized users.

Note: Oracle Automatic Storage Management (ASM) uses the concept of an instance for the memory and process components, but is not associated with a specific database.

Oracle Database Instance Configurations



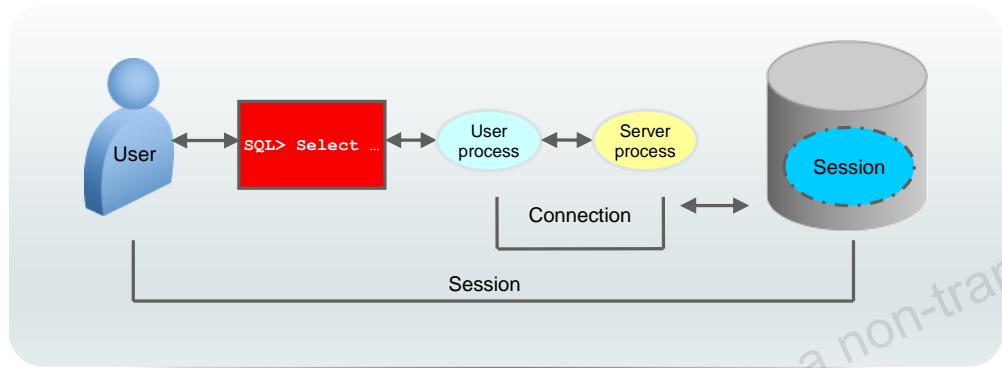
ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Each database instance is associated with only one database. If there are multiple databases on the same server, then there is a separate and distinct database instance for each database. A database instance cannot be shared. An Oracle Real Applications Cluster (RAC) database usually has multiple instances on separate servers for the same shared database. In this model, the same database is associated with each RAC instance, which meets the requirement that, at most, only one database is associated with an instance.

Connecting to the Database Instance

- Connection: Communication between a user process and an instance
- Session: Specific connection of a user to an instance through a user process



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

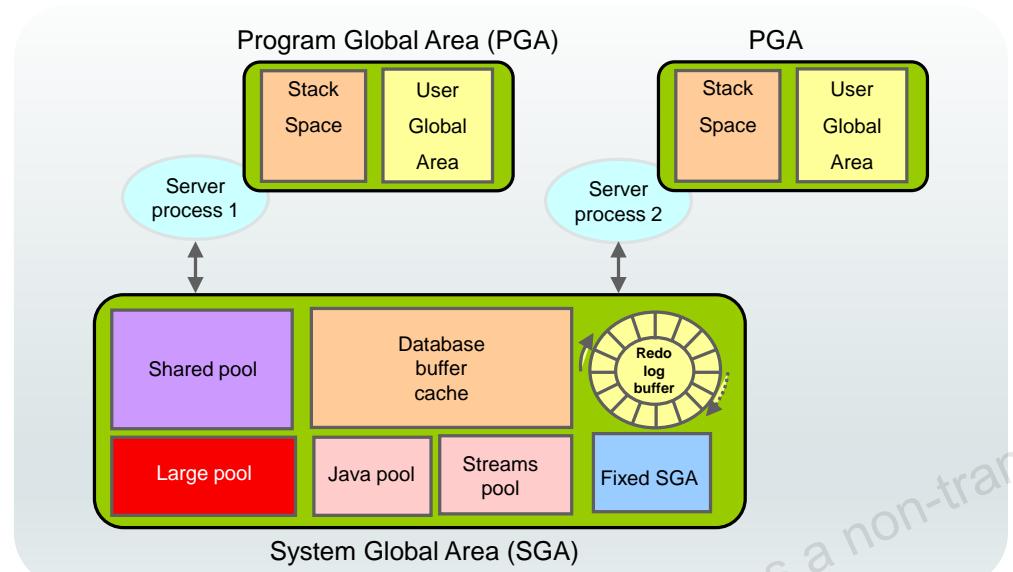
Connections and sessions are closely related to user processes but are very different in meaning.

A *connection* is a communication pathway between a user process and an Oracle Database instance. A communication pathway is established by using available interprocess communication mechanisms (on a computer that runs both the user process and Oracle Database) or network software (when different computers run the database application and Oracle Database and communicate through a network).

A *session* represents the state of a current user login to the database instance. For example, when a user starts SQL*Plus, the user must provide a valid username and password, and then a session is established for that user. A session lasts from the time a user connects until the user disconnects or exits the database application.

Multiple sessions can be created and exist concurrently for a single Oracle database user by using the same username. For example, a user with the username/password of HR/HR can connect to the same Oracle Database instance several times.

Oracle Database Memory Structures



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Database creates and uses memory structures for various purposes. For example, memory stores program code being run, data that is shared among users, and private data areas for each connected user.

Two basic memory structures are associated with an instance:

- **System Global Area (SGA):** Group of shared memory structures, known as SGA components, which contains data and control information for one Oracle Database instance. The SGA is shared by all server and background processes. Examples of data stored in the SGA include cached data blocks and shared SQL areas.
- **Program Global Area (PGA):** Memory regions that contain data and control information for a server or background process. A PGA is nonshared memory created by Oracle Database when a server or background process is started. Access to the PGA is exclusive to the server process. Each server process and background process has its own PGA.

The SGA is the memory area that contains data and control information for the instance. The SGA includes the following data structures:

- **Shared pool:** Caches various constructs that can be shared among users
- **Database buffer cache:** Caches blocks of data retrieved from the database
- **Redo log buffer:** Caches redo information (used for instance recovery) until it can be written to the physical redo log files stored on the disk
- **Large pool:** Optional area that provides large memory allocations for certain large processes, such as Oracle backup and recovery operations, and I/O server processes
- **Java pool:** Used for all session-specific Java code and data in Java Virtual Machine (JVM)
- **Streams pool:** Used by Oracle Streams to store information required by capture and apply
- **Fixed SGA:** An internal housekeeping area containing general information about the state of the database and the instance, and information communicated between processes

When you start the instance, the amount of memory allocated for the SGA is displayed.

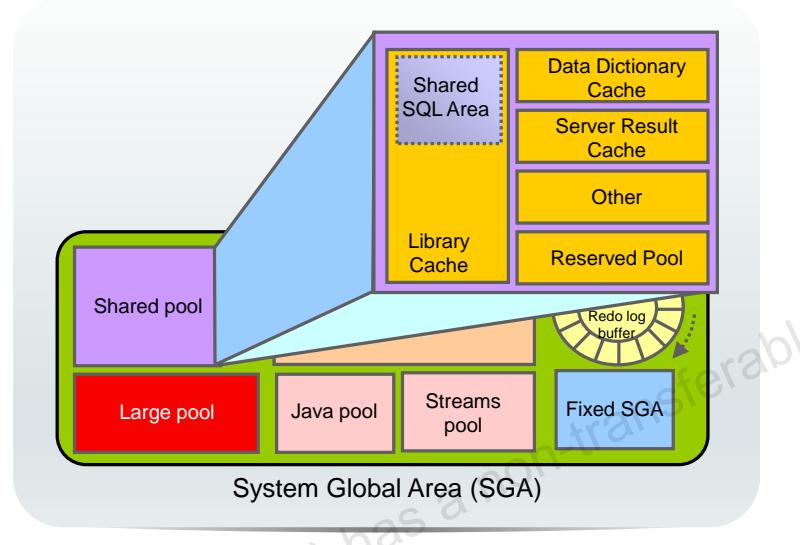
A Program Global Area (PGA) is a memory region that contains data and control information for each server process. An Oracle server process services a client's requests. Each server process has its own private PGA that is allocated when the server process is started. Access to the PGA is exclusive to that server process, and the PGA is read and written only by the Oracle code acting on its behalf. The PGA is divided into two major areas: stack space and the user global area (UGA).

With the dynamic SGA infrastructure, the sizes of the database buffer cache, the shared pool, the large pool, the Java pool, and the Streams pool can change without shutting down the instance.

The Oracle Database server uses initialization parameters to create and manage memory structures. The simplest way to manage memory is to allow the database to automatically manage and tune it for you. To do so (on most platforms), you only have to set a target memory size initialization parameter (`MEMORY_TARGET`) and a maximum memory size initialization parameter (`MEMORY_MAX_TARGET`).

Shared Pool

- Is a portion of the SGA
- Contains:
 - Library cache
 - Shared SQL area
 - Data dictionary cache
 - Server result cache



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The shared pool portion of the SGA contains the library cache, the data dictionary cache, the server result cache containing the SQL query result cache and the PL/SQL function result cache, buffers for parallel execution messages, and control structures.

The *data dictionary* is a collection of database tables and views containing reference information about the database, its structures, and its users. Oracle Database accesses the data dictionary frequently during SQL statement parsing. This access is essential to the continuing operation of Oracle Database.

The data dictionary is accessed so often by Oracle Database that two special locations in memory are designated to hold dictionary data. One area is called the *data dictionary cache*, also known as the row cache because it holds data as rows instead of buffers (buffers hold entire blocks of data). The other area in memory that holds dictionary data is the *library cache*. All Oracle Database user processes share these two caches for access to data dictionary information.

Oracle Database represents each SQL statement that it runs with a shared SQL area (as well as a private SQL area kept in the PGA). Oracle Database recognizes when two users are executing the same SQL statement and reuses the shared SQL area for those users.

A shared SQL area contains the parse tree and execution plan for a given SQL statement. Oracle Database saves memory by using one shared SQL area for SQL statements run multiple times, which often happens when many users run the same application.

When a new SQL statement is parsed, Oracle Database allocates memory from the shared pool to store in the shared SQL area. The size of this memory depends on the complexity of the statement.

Oracle Database processes PL/SQL program units (procedures, functions, packages, anonymous blocks, and database triggers) in much the same way it processes individual SQL statements. Oracle Database allocates a shared area to hold the parsed, compiled form of a program unit. Oracle Database allocates a private area to hold values specific to the session that runs the program unit, including local, global, and package variables (also known as package instantiation), and buffers for executing SQL. If more than one user runs the same program unit, then a single, shared area is used by all users, while all users maintain separate copies of their own private SQL areas, holding values specific to their own sessions.

Individual SQL statements contained in a PL/SQL program unit are processed just like other SQL statements. Despite their origins in a PL/SQL program unit, these SQL statements use a shared area to hold their parsed representations and a private area for each session that runs the statement.

The *server result cache* contains the *SQL query result cache* and *PL/SQL function result cache*, which share the same infrastructure. The server result cache contains result sets, not data blocks.

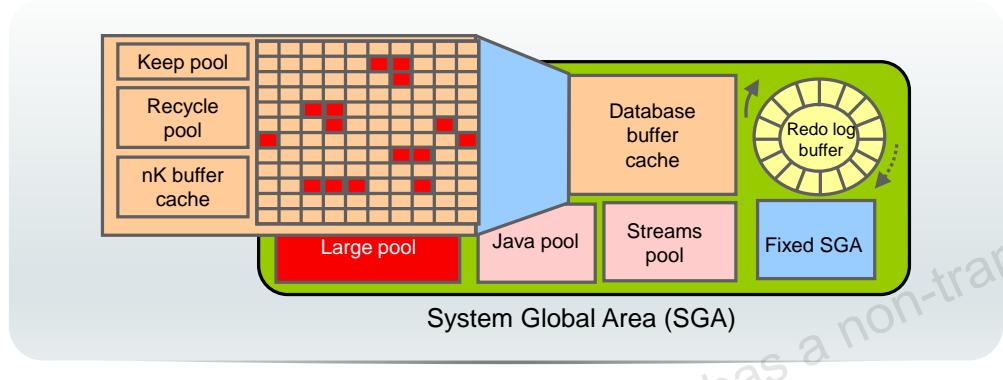
Results of queries and query fragments can be cached in memory in the SQL query result cache. The database server can then use cached results to answer future executions of these queries and query fragments. Because retrieving results from the SQL query result cache is faster than rerunning a query, frequently run queries experience a significant performance improvement when their results are cached.

A PL/SQL function is sometimes used to return the result of a computation whose inputs are one or several parameterized queries issued by the function. In some cases, these queries access data that changes very infrequently compared to the frequency of calling the function. You can include syntax in the source text of a PL/SQL function to request that its results be cached in the PL/SQL function result cache and (to ensure correctness) that the cache be purged when tables in a list of tables experience data manipulation language (DML).

The *reserved pool* is a memory area in the shared pool that Oracle Database can use to allocate large contiguous chunks of memory.

Database Buffer Cache

- Is part of the SGA
- Holds copies of data blocks that are read from data files
- Is shared by all concurrent users



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The database buffer cache is the portion of the SGA that holds block images read from data files or constructed dynamically to satisfy the read consistency model. All users who are concurrently connected to the instance share access to the database buffer cache.

The first time an Oracle Database user process requires a particular piece of data, it searches for the data in the database buffer cache. If the process finds the data already in the cache (a cache hit), it can read the data directly from memory. If the process cannot find the data in the cache (a cache miss), it must copy the data block from a data file on disk into a buffer in the cache before accessing the data. Accessing data through a cache hit is faster than accessing data through a cache miss.

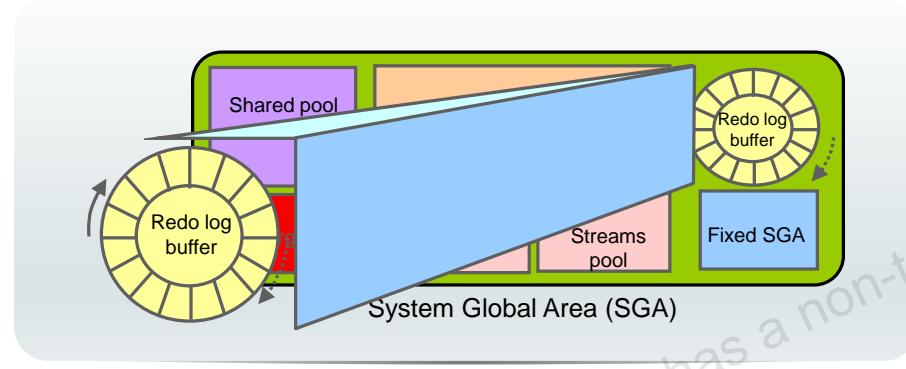
The buffers in the cache are managed by a complex algorithm that uses a combination of least recently used (LRU) lists and touch count. The LRU helps to ensure that the most recently used blocks tend to stay in memory to minimize disk access.

The keep buffer pool and the recycle buffer pool are used for specialized buffer pool tuning. The keep buffer pool is designed to retain buffers in memory longer than the LRU would normally retain them. The recycle buffer pool is designed to flush buffers from memory faster than the LRU normally would.

Additional buffer caches can be configured to hold blocks of a size that is different from the default block size.

Redo Log Buffer

- Is a circular buffer in the SGA
- Holds information about changes made to the database
- Contains redo entries that have the information to redo changes made by operations such as DML and DDL



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

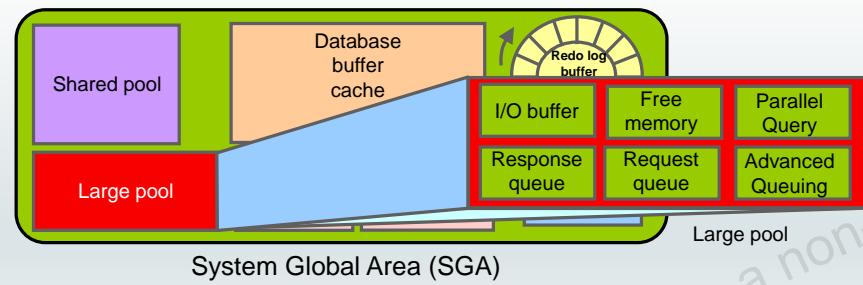
The redo log buffer is a circular buffer in the SGA that holds information about changes made to the database. This information is stored in redo entries. Redo entries contain the information necessary to reconstruct (or redo) changes that are made to the database by DML, DDL, or internal operations. Redo entries are used for database recovery, if necessary.

As the server process makes changes to the buffer cache, redo entries are generated and written to the redo log buffer in the SGA. The redo entries take up continuous, sequential space in the buffer. The log writer background process writes the redo log buffer to the active redo log file (or group of files) on disk.

Large Pool

Provides large memory allocations for:

- Session memory for the shared server and the Oracle XA interface
- I/O server processes
- Oracle Database backup and restore operations



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The database administrator can configure an optional memory area called the *large pool* to provide large memory allocations for:

- Session memory for the shared server and the Oracle XA interface (used where transactions interact with multiple databases)
- I/O server processes
- Oracle Database backup and restore operations
- Parallel Query operations
- Advanced Queuing memory table storage

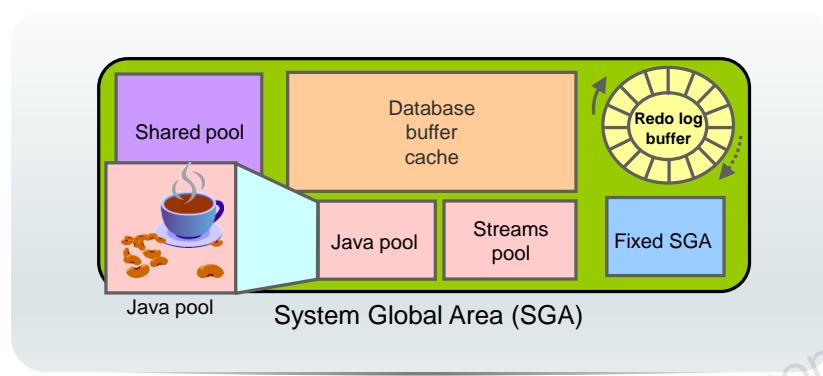
By allocating session memory from the large pool for shared server, Oracle XA, or parallel query buffers, Oracle Database can use the shared pool primarily for caching shared SQL and avoid the performance overhead that is caused by shrinking the shared SQL cache.

In addition, the memory for Oracle Database backup and restore operations, for I/O server processes and for parallel buffers, is allocated in buffers of a few hundred kilobytes. The large pool is better able to satisfy such large memory requests than the shared pool.

The large pool is not managed by a least recently used (LRU) list.

Java Pool

Java pool memory is used to store all session-specific Java code and data in the JVM.



ORACLE®

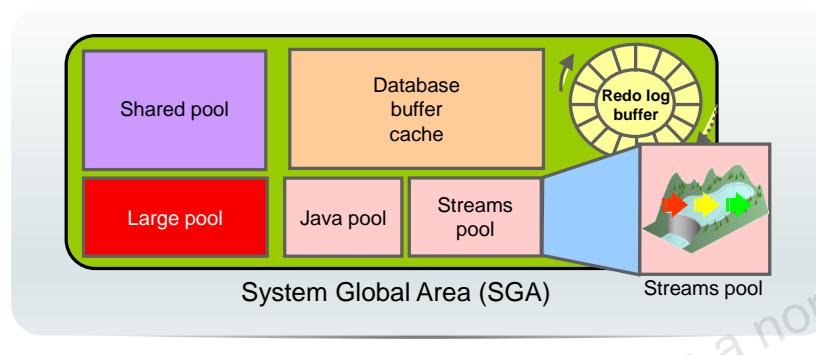
Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Java pool memory is used to store all session-specific Java code and data in the Java Virtual Machine (JVM). Java pool memory is used in different ways, depending on the mode in which Oracle Database is running.

Streams Pool

Streams pool memory is used exclusively by Oracle Streams to:

- Store buffered queue messages
- Provide memory for Oracle Streams processes



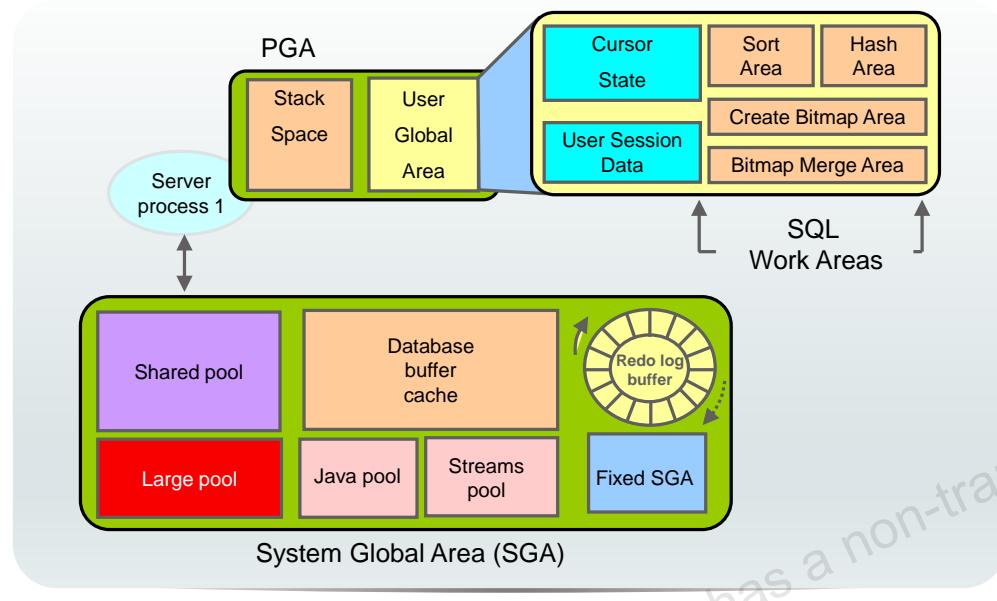
ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The Streams pool is used exclusively by Oracle Streams. The Streams pool stores buffered queue messages and provides memory for Oracle Streams capture processes and apply processes.

Unless you specifically configure it, the size of the Streams pool starts at zero. The pool size grows dynamically as needed when Oracle Streams is used.

Program Global Area (PGA)



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The Program Global Area (PGA) is a private memory region containing data and control information for a server process. Each server process has a distinct PGA. Access to it is exclusive to that server process, and it is read only by Oracle code acting on behalf of it. It is not available for developer's code.

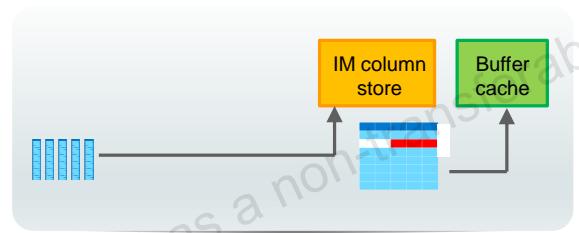
Every PGA contains stack space. In a dedicated server environment, each user connecting to the database instance has a separate server process. For this type of connection, the PGA contains a subdivision of memory known as the User Global Area (UGA). The UGA is composed of:

- Cursor area for storing runtime information on cursors
- User session data storage area for control information about a session
- SQL working areas for processing SQL statements consisting of:
 - A sort area for functions that order data such as ORDER BY and GROUP BY
 - A hash area for performing hash joins of tables
 - A create bitmap area used in bitmap index creation common to data warehouses
 - A bitmap merge area used for resolving bitmap index plan execution

In a shared server environment, multiple client users share the server process. In this model, the UGA is moved into the SGA (shared pool or large pool if configured), leaving the PGA with only stack space.

In-Memory Column Store: Introduction

- Instant query response:
 - Faster queries on very large tables on any columns (100x)
 - Use of scans, joins, and aggregates
 - Without indexes
 - Best suited for analytics: few columns, many rows
- Faster DML: Removal of most analytics indexes (3 to 4x)
- Full application transparency
- Easy setup:
 - In-memory column store configuration
 - Segment attributes



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The In-Memory Column Store feature enables objects (tables, partitions, and other types) to be stored in memory in a new format known as the *columnar format*. This format enables scans, joins, and aggregates to perform much faster than the traditional on-disk format, thus providing fast reporting and DML performance for both OLTP and DW environments. This is particularly useful for analytic applications that operate on few columns returning many rows rather than for OLTP that operates on few rows returning many columns. The DBA must define the segments that are to be populated into the in-memory column store (IM column store), such as hot tables, partitions, and, more precisely, the more frequently accessed columns.

The in-memory columnar format does not replace the on-disk or buffer cache format. It is a consistent copy of a table or of some columns of a table converted to the new columnar format that is independent of the disk format and only available in memory. Because of this independence, applications are able to transparently use this option without any changes. For the data to be converted into the new columnar format, a new pool is requested in the SGA. The pool is the IM column store.

If sufficient space is allocated for the IM column store, a query that accesses objects that are candidates to be populated into the IM column store performs much faster. The improved performance allows ad hoc analytic queries to be executed directly on the real-time transaction data without impacting the existing workload.

There are three main advantages:

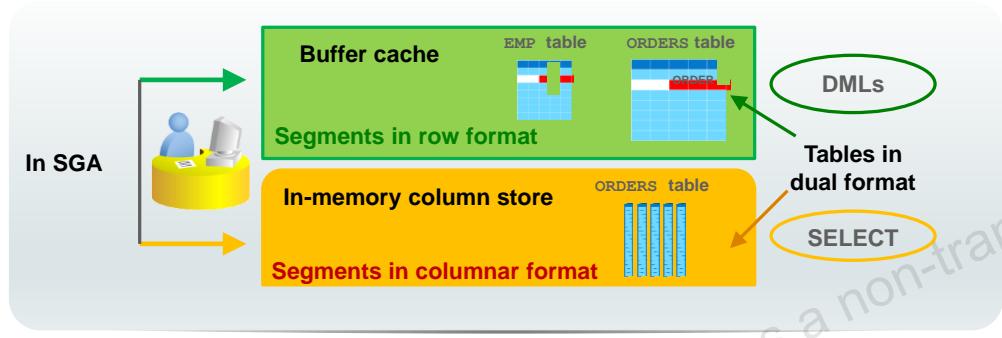
- Queries run a lot faster: All data can be populated in memory in a compressed columnar format. No index is required and used. Queries run at least 100 times faster than when fetching data from the buffer cache, thanks to the columnar compressed format.
- DMLs are faster: Analytics indexes can be eliminated by being replaced by scans of the IM column store representation of the table.
- Arbitrary ad hoc queries run with good performance because the table behaves as if all columns are indexed.

Note: The In-Memory Column Store feature is included with the Oracle Database In-Memory option.

Refer to the *Oracle Database Administrator's Guide* for detailed information about this feature.

In-Memory Column Store: Overview

- A pool in the SGA: In-Memory column store
 - Segments populated into the IM column store are converted into a columnar format.
 - In-Memory segments are transactionally consistent with the buffer cache.
- Only one segment on disk and in row format



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The in-memory columnar format does not replace the on-disk or buffer cache format. This means that when a segment, such as a table or a partition, is populated into the IM column store, the on-disk format segment is automatically converted into a columnar format and optionally compressed. The columnar format is a pure in-memory format. There is no columnar format storage on disk. It never causes additional writes to disk and therefore does not require any logging or undo space.

All data is stored on disk in the traditional row format.

Moreover, the columnar format of a segment is a transaction-consistent copy of the segment either on disk or in the buffer cache. Transaction consistency between the two pools is maintained.

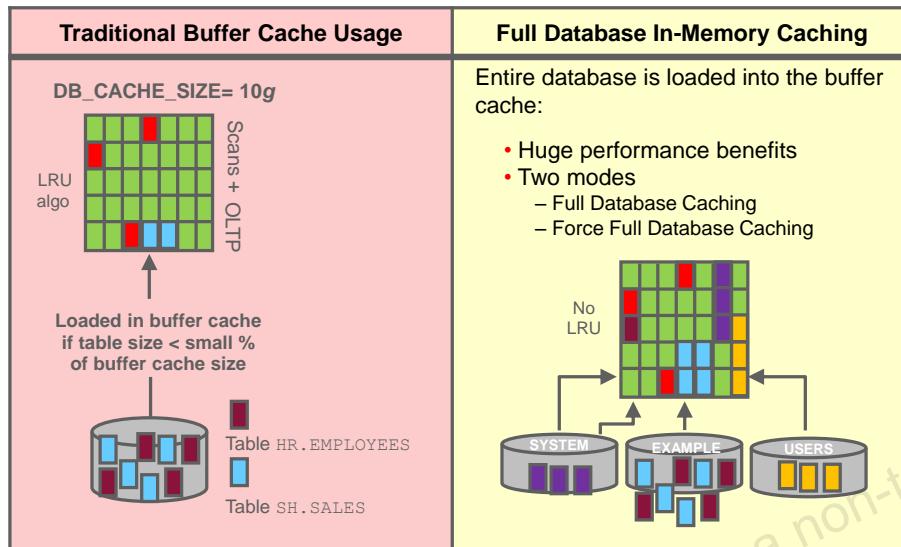
If sufficient space is allocated to the IM column store in SGA, a query that accesses objects that are populated into the IM column store performs much faster. The improved performance allows more ad hoc analytic queries to be executed directly on real-time transaction data without impacting the existing workload. A lack of IM column store space does not prevent statements from executing against tables that could have been populated into the IM column store.

The DBA must decide, according to the types of queries and DMLs that are executed against the segments, which segments should be defined as non-in-memory segments and which should be defined as in-memory segments. The DBA can also define more precisely which columns are good candidates for IM column store:

- **In row format exclusively:** The segments that are being frequently accessed by OLTP-style queries, which operate on few rows returning many columns, are good candidates for the buffer cache. These segments should not necessarily be defined as in-memory segments and should be sent to the buffer cache only.
- **In dual format simultaneously:** The segments that are being frequently accessed by analytical-style queries, which operate on many rows returning a few columns, are good candidates for IM column store. If a segment is defined as an in-memory segment but has some columns that are defined as non-in-memory columns, the queries that select any non-in-memory columns are sent to the buffer cache and those selecting in-memory columns only are sent to the IM column store. Any fetch-by-rowid is performed on the segment through the buffer cache.

Any DML performed on these objects is executed via the buffer cache.

Full Database In-Memory Caching



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

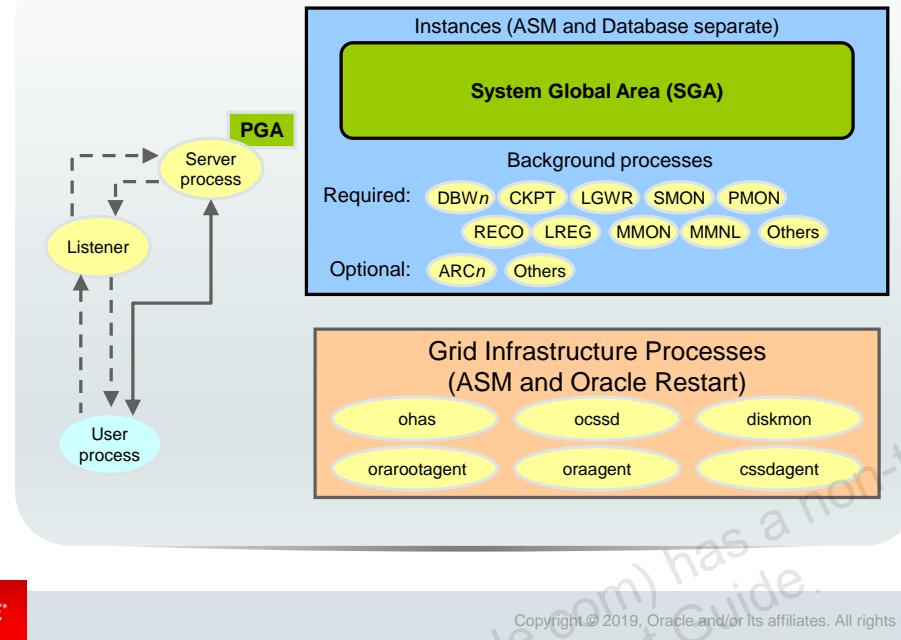
The current algorithm for table scans loads a table into the buffer cache only when the table size is less than a small percent of the buffer cache size. For very large tables, the database uses a direct path read, which loads blocks directly into the PGA and bypasses the SGA, to avoid flooding the buffer cache. The DBA must explicitly declare small lookup tables, which are accessed frequently, as CACHE to load data into memory and avoid bypassing the SGA. This clause indicates that the blocks retrieved for these tables are placed at the most recently used end of the least recently used (LRU) list in the buffer cache when a full table scan is performed.

The Full Database In-memory Caching feature enables an entire database to be cached in memory when the database size (sum of all data files, SYSTEM tablespace, LOB CACHE files minus SYSAUX, TEMP) is smaller than the buffer cache size. Caching and running a database from memory leads to huge performance benefits. Two modes can be used:

- **Full Database Caching:** Implicit default and automatic mode in which an internal calculation determines if the database can be fully cached for an instance. NOCACHE LOBs are not cached in Full Database Caching. But in Force Full Database Caching mode, even NOCACHE LOBs are cached.
- **Force Full Database Caching:** Neither Full Database Caching nor Force Full Database Caching forces or prefetches data into memory. Workload must access the data first for them to be cached. It considers the entire database as eligible to be completely cached in the buffer cache. This mode requires the DBA to execute the ALTER DATABASE FORCE FULL DATABASE CACHING command. This mode takes precedence over Full Database Caching mode. To revert to traditional caching, use the ALTER DATABASE NO FORCE FULL DATABASE CACHING command.

Refer to *Oracle Database Administrators Guide* and *Oracle Database Performance Tuning Guide* for detailed information about this feature.

Process Structures



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Server Processes

Oracle Database creates server processes to handle the requests of user processes connected to the instance. The user process represents the application or tool that connects to the Oracle database. It may be on the same machine as the Oracle database, or it may exist on a remote client and use a network to reach the Oracle database. The user process first communicates with a listener process that creates a server process in a dedicated environment.

Server processes created on behalf of each user's application can perform one or more of the following:

- Parse and run SQL statements issued through the application.
- Read necessary data blocks from data files on disk into the shared database buffers of the SGA (if the blocks are not already present in the SGA).
- Return results in such a way that the application can process the information.

Background Processes

To maximize performance and accommodate many users, a multiprocess Oracle Database system uses some additional Oracle Database processes called *background processes*. An Oracle Database instance can have many background processes.

The background processes commonly seen in non-RAC, non-ASM environments can include the following:

- Database Writer process (DBW n)
- Log Writer process (LGWR)
- Checkpoint process (CKPT)
- System monitor process (SMON)
- Process monitor process (PMON)
- Recoverer process (RECO)
- Listener registration process (LREG)
- Manageability monitor process (MMON)
- Manageability monitor lite process (MMNL)
- Job queue coordinator (CJQ0)
- Job slave processes (Jnnn)
- Archiver processes (ARC n)
- Queue monitor processes (QMNN)

Other background processes may be found in more advanced configurations such as RAC. See the V\$BGPROCESS view for more information on the background processes.

Some background processes are created automatically when an instance is started, whereas others are started as required.

Other process structures are not specific to a single database, but rather can be shared among many databases on the same server. The Grid Infrastructure and networking processes fall into this category.

Oracle Grid Infrastructure processes on Linux and UNIX systems include the following:

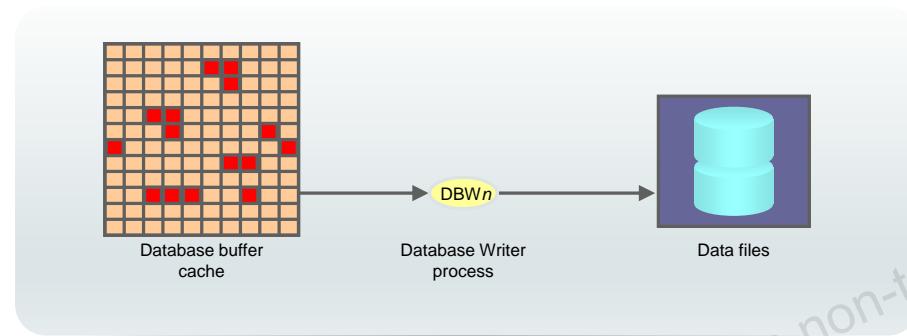
- `ohasd` (Oracle High Availability Service daemon): Is responsible for starting Oracle Clusterware processes
- `ocssd`: Cluster Synchronization Service daemon
- `diskmon` (Disk Monitor daemon): Is responsible for input and output fencing for Oracle Exadata Storage
- `cssdagent`: Starts, stops, and checks the status of the CSS daemon, `ocssd`
- `oraagent`: Extends Clusterware to support Oracle-specific requirements and complex resources
- `orarootagent`: Is a specialized Oracle agent process that helps manage resources owned by root, such as the network

Note: For a more detailed list of the background processes, consult the *Oracle Database Reference* guide.

Database Writer Process (DBWn)

Writes modified (dirty) buffers in the database buffer cache to disk:

- Asynchronously while performing other processing
- To advance the checkpoint



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The Database Writer process (DBW n) writes the contents of buffers to data files. The DBW n processes are responsible for writing modified (dirty) buffers in the database buffer cache to disk. Although one Database Writer process (DBW0) is adequate for most systems, you can configure additional processes to improve write performance if your system modifies data heavily. The additional processes are named DBW1 through DBW9, DBWa through DBWz, and BW36-BW99. These additional DBW n processes are not useful on uniprocessor systems.

When a buffer in the database buffer cache is modified, it is marked dirty and added to the head of the checkpoint queue that is kept in system change number (SCN) order. This order, therefore, matches the order of redo that is written to the redo logs for these changed buffers. When the number of available buffers in the buffer cache falls below an internal threshold (to the extent that server processes find it difficult to obtain available buffers), DBW n writes nonfrequently used buffers to the data files from the tail of the LRU list so that processes can replace buffers when they need them. DBW n also writes from the tail of the checkpoint queue to keep the checkpoint advancing.

The SGA contains a memory structure that has the redo byte address (RBA) of the position in the redo stream where recovery should begin in case of an instance failure. This structure acts as a pointer into the redo and is written to the control file by the CKPT process once every three seconds. Because the DBW n writes dirty buffers in SCN order, and because the redo is in SCN order, every time DBW n writes dirty buffers from the LRU list, it also advances the pointer held in the SGA memory structure so that instance recovery (if required) begins reading the redo from approximately the correct location and avoids unnecessary I/O. This is known as *incremental checkpointing*.

Note: There are other cases when DBW n may write (for example, when tablespaces are made read-only or are placed offline). In such cases, no incremental checkpoint occurs because dirty buffers belonging only to the corresponding data files are written to the database unrelated to the SCN order.

The LRU algorithm keeps more frequently accessed blocks in the buffer cache to minimize disk reads. A CACHE option can be placed on tables to help retain blocks even longer in memory.

The DB_WRITER_PROCESSES initialization parameter specifies the number of DBW n processes. The maximum number of Database Writer processes is 100. If it is not specified by the user during startup, Oracle Database determines how to set DB_WRITER_PROCESSES based on the number of CPUs and processor groups.

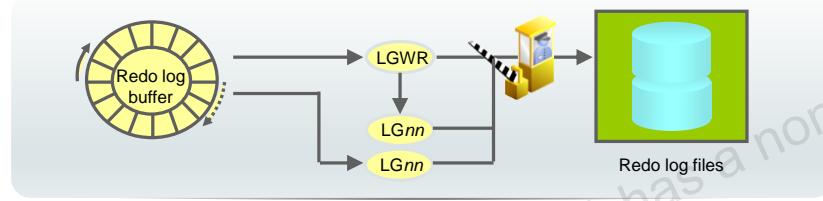
The DBW n process writes dirty buffers to disk under the following conditions:

- When a server process cannot find a clean reusable buffer after scanning a threshold number of buffers, it signals DBW n to write. DBW n writes dirty buffers to disk asynchronously while performing other processing.
- DBW n writes buffers to advance the checkpoint, which is the position in the redo thread (log) from which instance recovery begins. This log position is determined by the oldest dirty buffer in the buffer cache.

In all cases, DBW n performs batched (multiblock) writes to improve efficiency. The number of blocks written in a multiblock write varies by operating system.

Log Writer Process (LGWR)

- Writes the redo log buffer to a redo log file on disk:
 - When a user process commits a transaction
 - When an online redo log switch occurs
 - When the redo log buffer is one-third full or contains 1 MB of buffered data
 - Before a DBW n process writes modified buffers to disk
 - When three seconds have passed since the last write
- Serves as coordinator of LG n processes and ensures correct order for operations that must be ordered



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The Log Writer process (LGWR) is responsible for redo log buffer management by writing the redo log buffer entries to a redo log file on disk. LGWR writes all redo entries that have been copied into the buffer since the last time it wrote.

LGWR starts and coordinates multiple helper processes that concurrently perform some of the work. LGWR handles the operations that are very fast or must be coordinated and delegates operations to the LG n n that could benefit from concurrent operations, primarily writing the redo from the log buffer to the redo log file and posting the completed write to the foreground process that is waiting.

Because LG n n processes work concurrently and certain operations must be performed in order, LGWR forces ordering so that even if the writes complete out of order, the posting to the foreground processes will be in the correct order.

The redo log buffer is a circular buffer. When LGWR writes redo entries from the redo log buffer to a redo log file, server processes can then copy new entries over the entries in the redo log buffer that have been written to disk. LGWR normally writes fast enough to ensure that space is always available in the buffer for new entries, even when access to the redo log is heavy. LGWR writes one contiguous portion of the buffer to disk.

LGWR writes:

- When a user process commits a transaction
- When an online redo log switch occurs
- When the redo log buffer is one-third full or contains 1 MB of buffered data
- Before a DBW n process writes modified buffers to disk (if necessary)
- When three seconds have passed since the last write to log files

Before DBW n can write a modified buffer, all redo records that are associated with the changes to the buffer must be written to disk (the write-ahead protocol). If DBW n finds that some redo records have not been written, it signals LGWR to write the redo records to disk and waits for LGWR to complete writing the redo log buffer before it can write out the data buffers. LGWR writes to the current log group. If one of the files in the group is damaged or unavailable, LGWR continues writing to other files in the group and logs an error in the LGWR trace file and in the system alert log. If all files in a group are damaged, or if the group is unavailable because it has not been archived, LGWR cannot continue to function.

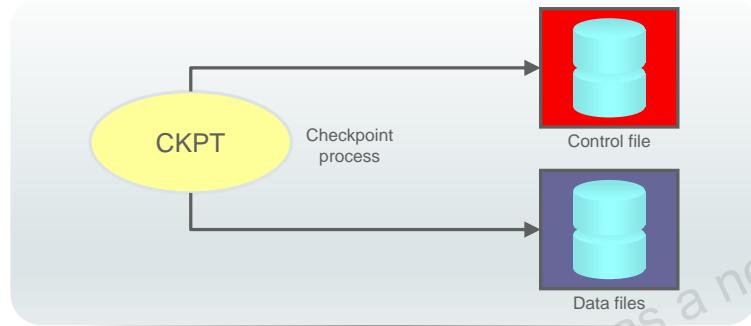
When a user issues a `COMMIT` statement, LGWR puts a commit record in the redo log buffer and writes it to disk immediately, along with the transaction's redo entries. The corresponding changes to data blocks are deferred until it is more efficient to write them. This is called a *fast commit mechanism*. The atomic write of the redo entry containing the transaction's commit record is the single event that determines whether the transaction has committed. Oracle Database returns a success code to the committing transaction, although the data buffers have not yet been written to disk.

If more buffer space is needed, LGWR sometimes writes redo log entries before a transaction is committed. These entries become permanent only if the transaction is later committed. When a user commits a transaction, the transaction is assigned an SCN, which Oracle Database records along with the transaction's redo entries in the redo log. SCNs are recorded in the redo log so that recovery operations can be synchronized in Real Application Clusters and distributed databases.

In times of high activity, LGWR can write to the redo log file by using group commits. For example, assume a user commits a transaction. LGWR must write the transaction's redo entries to disk. As this happens, other users issue `COMMIT` statements. However, LGWR cannot write to the redo log file to commit these transactions until it has completed its previous write operation. After the first transaction's entries are written to the redo log file, the entire list of redo entries of waiting transactions (not yet committed) can be written to disk in one operation, requiring less I/O than transaction entries handled individually. Therefore, Oracle Database minimizes disk I/O and maximizes the performance of LGWR. If requests to commit continue at a high rate, every write (by LGWR) from the redo log buffer can contain multiple commit records.

Checkpoint Process (CKPT)

- Records checkpoint information in:
 - The Control file
 - Each data file header
- Signals DBW n to write blocks to disk



ORACLE®

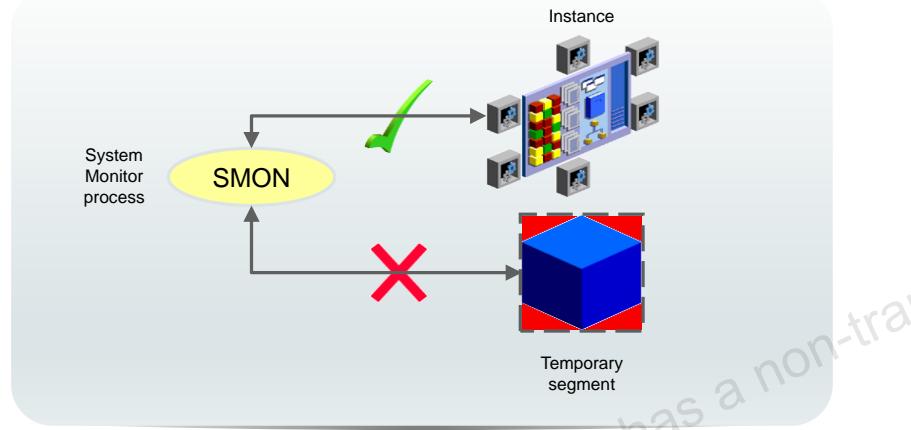
Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

A *checkpoint* is a data structure that defines a system change number (SCN) in the redo thread of a database. Checkpoints are recorded in the control file and in each data file header. They are a crucial element of recovery.

When a checkpoint occurs, Oracle Database must update the headers of all data files to record the details of the checkpoint. This is done by the CKPT process. The CKPT process does not write blocks to disk; DBW n always performs that work. The SCNs recorded in the file headers guarantee that all changes made to database blocks before that SCN have been written to disk.

System Monitor Process (SMON)

- Performs recovery at instance startup
- Cleans up unused temporary segments



ORACLE®

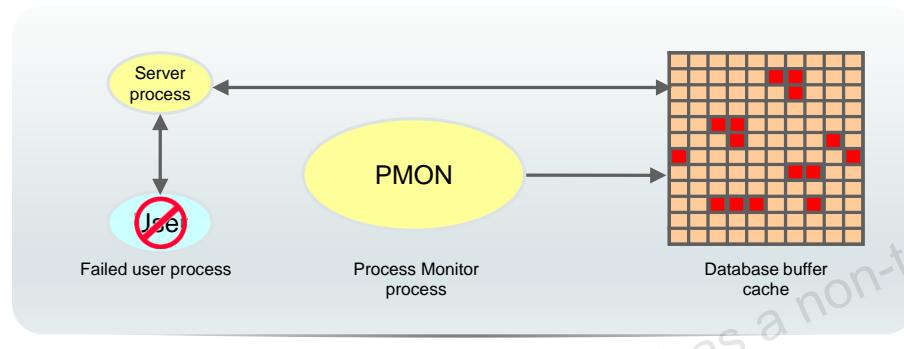
Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The System Monitor process (SMON) performs recovery at instance startup if necessary. SMON is also responsible for cleaning up temporary segments that are no longer in use. If any terminated transactions were skipped during instance recovery because of file-read or offline errors, SMON recovers them when the tablespace or file is brought back online.

SMON checks regularly to see whether the process is needed. Other processes can call SMON if they detect a need for it.

Process Monitor Process (PMON)

- Performs process recovery when a user process fails
 - Cleans up the database buffer cache
 - Frees resources that are used by the user process
- Monitors sessions for idle session timeout



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

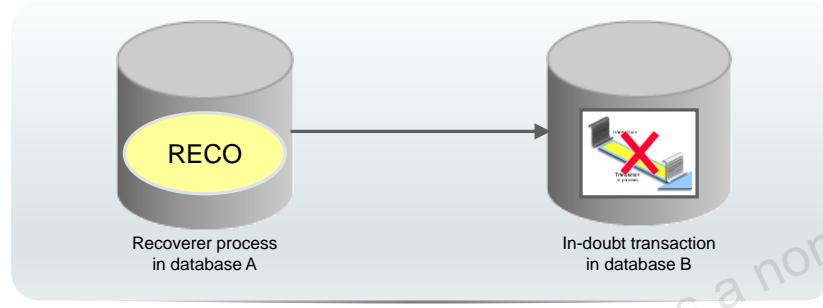
The Process Monitor process (PMON) performs process recovery when a user process fails. PMON is responsible for cleaning up the database buffer cache and freeing resources that the user process was using. For example, it resets the status of the active transaction table, releases locks, and removes the process ID from the list of active processes.

PMON periodically checks the status of dispatcher and server processes and restarts any that have stopped running (but not any that Oracle Database has terminated intentionally).

Like SMON, PMON checks regularly to see whether it is needed; it can be called if another process detects the need for it.

Recoverer Process (RECO)

- Used with the distributed database configuration
- Automatically connects to other databases involved in in-doubt distributed transactions
- Automatically resolves all in-doubt transactions
- Removes any rows that correspond to in-doubt transactions



ORACLE®

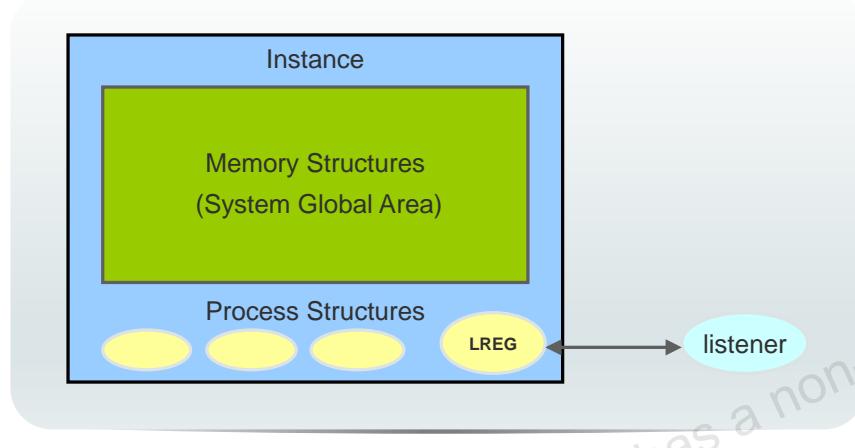
Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The Recoverer process (RECO) is a background process used with the distributed database configuration that automatically resolves failures involving distributed transactions. The RECO process of an instance automatically connects to other databases involved in an in-doubt distributed transaction. When the RECO process reestablishes a connection between involved database servers, it automatically resolves all in-doubt transactions, removing from each database's pending transaction table any rows that correspond to the resolved in-doubt transactions.

If the RECO process fails to connect with a remote server, RECO automatically tries to connect again after a timed interval. However, RECO waits an increasing amount of time (growing exponentially) before it attempts another connection.

Listener Registration Process (LREG)

Registers information about the database instance and dispatcher processes with Oracle Net Listener



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The Listener Registration process (LREG) registers information about the database instance and dispatcher processes with the Oracle Net Listener. LREG provides the listener with the following information:

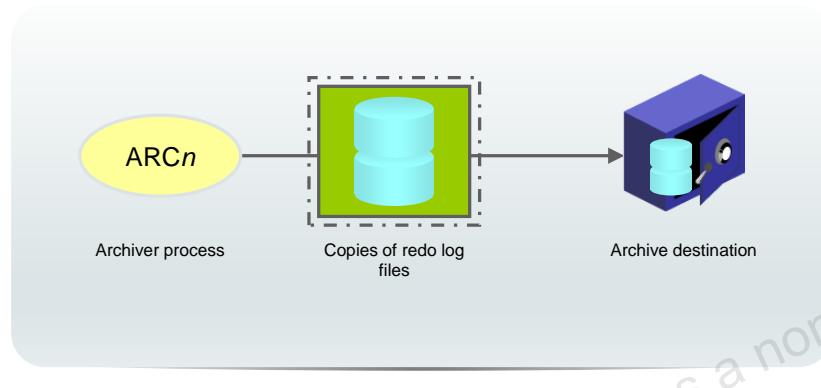
- Names of the database services
- Name of the database instance associated with the services, and its current and maximum load
- Service handlers (dispatchers and dedicated servers) available for the instance, including their type, protocol addresses, and current and maximum load

When the instance starts, LREG attempts to connect to the listener. If the listener is running, LREG passes information to it. If the listener is not running, LREG periodically attempts to connect to it. It may take up to 60 seconds for LREG to register the database instance with the listener after the listener has started.

You can use the `ALTER SYSTEM REGISTER` command to immediately initiate service registration after starting the listener.

Archiver Processes (ARC n)

- Copy redo log files to a designated storage device after a log switch has occurred
- Can collect transaction redo data and transmit that data to standby destinations



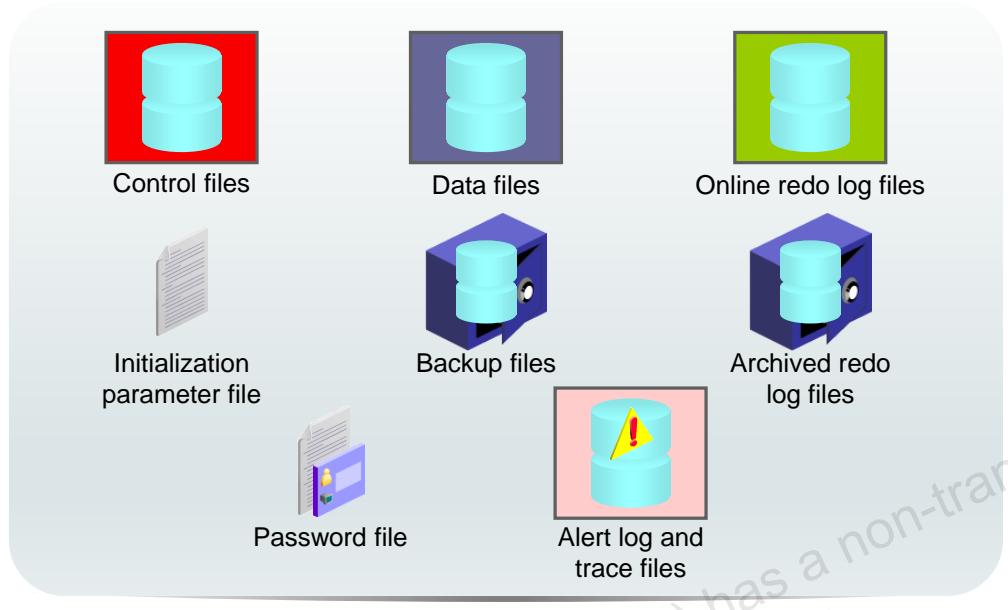
ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The Archiver processes (ARC n) copy redo log files to a designated storage device after a log switch has occurred. ARC n processes are present only when the database is in ARCHIVELOG mode and automatic archiving is enabled.

If you anticipate a heavy workload for archiving (such as during bulk loading of data), you can increase the maximum number of Archiver processes. There can also be multiple archive log destinations. It is recommended that there be at least one Archiver process for each destination. The default is to have four Archiver processes.

Database Storage Architecture



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The files that comprise an Oracle database are as follows:

- **Control files:** Each database has one unique control file that contains data about the database itself (that is, physical database structure information). Multiple copies may be maintained to protect against total loss. It can also contain metadata related to backups. The control file is critical to the database. Without the control file, the database cannot be opened.
- **Data files:** They contain the user or application data of the database, as well as metadata and the data dictionary.
- **Online redo log files:** They allow for instance recovery of the database. If the database server crashes and does not lose any data files, the instance can recover the database with the information in these files.

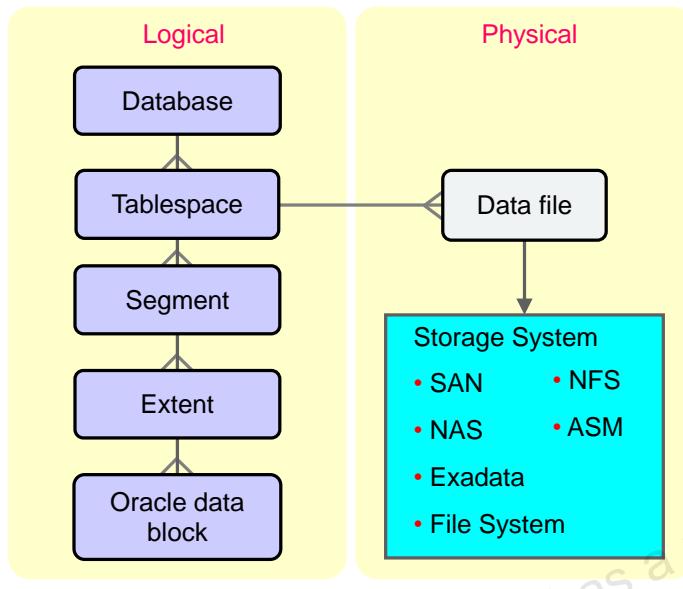
The following additional files are used during the operation of the database:

- **Initialization parameter file:** Is used to define how the instance is configured when it starts up
- **Password file:** Allows users using the SYSDBA, SYSOPER, SYSBACKUP, SYSDG, SYSKM, and SYSASM roles to connect remotely to the instance and perform administrative tasks
- **Backup files:** Are used for database recovery. You typically restore a backup file when a media failure or user error has damaged or deleted the original file.

- **Archived redo log files:** Contain an ongoing history of the data changes (redo) that are generated by the instance. Using these files and a backup of the database, you can recover a lost data file. That is, archive logs enable the recovery of restored data files.
- **Trace files:** Each server and background process can write to an associated trace file. When an internal error is detected by a process, the process dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database administrator, whereas other information is for Oracle Support Services.
- **Alert log file:** These are special trace entries. The alert log of a database is a chronological log of messages and errors. Oracle recommends that you review the alert log periodically.

Note: Parameter, password, alert, and trace files are covered in other lessons.

Logical and Physical Database Structures



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The database has logical structures and physical structures.

Databases, Tablespaces, and Data Files

The relationship among databases, tablespaces, and data files is illustrated in the slide. Each database is logically divided into two or more tablespaces. One or more data files are explicitly created for each tablespace to physically store the data of all segments in a tablespace. If it is a TEMPORARY tablespace, it has a temporary file instead of a data file. A tablespace's data file can be physically stored on any supported storage technology.

Tablespaces

A database is divided into logical storage units called *tablespaces*, which group related logical structures or data files together. For example, tablespaces commonly group all of an application's segments to simplify some administrative operations.

Data Blocks

At the finest level of granularity, an Oracle database's data is stored in *data blocks*. One data block corresponds to a specific number of bytes of physical space on the disk. A data block size is specified for each tablespace when it is created. A database uses and allocates free database space in Oracle data blocks.

Extents

The next level of logical database space is an *extent*. An extent is a specific number of contiguous Oracle data blocks (obtained in a single allocation) that are used to store a specific type of information. Oracle data blocks in an extent are logically contiguous but can be physically spread out on disk because of RAID striping and file system implementations.

Segments

The level of logical database storage above an extent is called a *segment*. A segment is a set of extents allocated for a certain logical structure. For example:

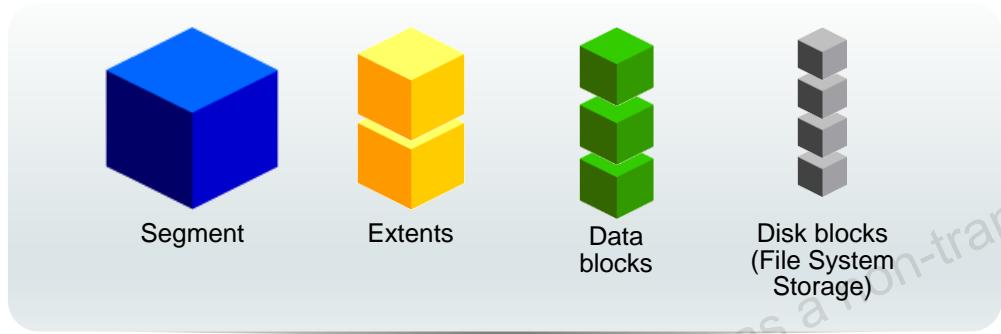
- **Data segments:** Each nonclustered, non-index-organized table has a data segment, except external tables, global temporary tables, and partitioned tables (in which each table has one or more segments). All the table's data is stored in the extents of its data segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
- **Index segments:** Each index has an index segment that stores all its data. For a partitioned index, each partition has an index segment.
- **Undo segments:** One UNDO tablespace is created for each database instance. This tablespace contains numerous undo segments to temporarily store undo information. The information in an undo segment is used to generate read-consistent database information and, during database recovery, roll back uncommitted transactions for users.
- **Temporary segments:** Temporary segments are created by the Oracle database when a SQL statement needs a temporary work area to complete execution. When the statement finishes execution, the temporary segment's extents are returned to the database for future use. Specify either a default temporary tablespace for every user or a default temporary tablespace that is used databasewide.

Note: There are other types of segments not listed here. There are also schema objects such as views, packages, and triggers that are not considered segments even though they are database objects. A segment owns its respective disk space allocation. The other objects exist as rows stored in a system metadata segment.

The Oracle Database server dynamically allocates space. When the existing extents of a segment are full, additional extents are added. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on the disk, and they can come from different data files belonging to the same tablespace.

Segments, Extents, and Blocks

- Segments exist in a tablespace.
- Segments are collections of extents.
- Extents are collections of data blocks.
- Data blocks are mapped to disk blocks.



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

A subset of database objects, such as tables and indexes, are stored as segments in tablespaces. Each segment contains one or more extents. An extent consists of contiguous data blocks, which means that each extent can exist only in one data file. Data blocks are the smallest unit of I/O in the database.

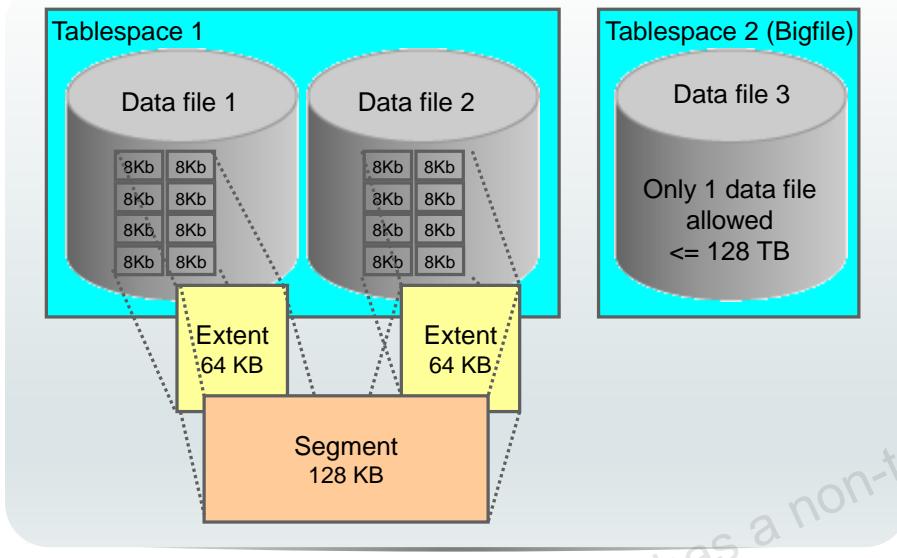
When the database requests a set of data blocks from the operating system (OS), the OS maps this to an actual file system or disk block on the storage device. Because of this, you do not need to know the physical address of any of the data in your database. This also means that a data file can be striped or mirrored on several disks.

The size of the data block can be set at the time of database creation. The default size of 8 KB is adequate for most databases. If your database supports a data warehouse application that has large tables and indexes, a larger block size may be beneficial.

If your database supports a transactional application in which reads and writes are random, specifying a smaller block size may be beneficial. The maximum block size depends on your OS. The minimum Oracle block size is 2 KB; it should rarely (if ever) be used.

You can have tablespaces with a nonstandard block size. For details, see *Oracle Database Administrator's Guide*.

Tablespaces and Data Files



A database is divided into *tablespaces*, which are logical storage units that can be used to group related logical structures. One or more data files are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace.

The graphic in the slide illustrates tablespace 1, composed of two data files. A segment of 128 KB size, composed of two extents, is spanning the two data files. The first extent of size 64 KB is in the first data file and the second extent, also of size 64 KB, is in the second data file. Both extents are formed from contiguous 8 KB Oracle blocks.

Note: You can also create bigfile tablespaces, which have only one file that is often very large. The file may be of any size up to the maximum that the row ID architecture permits. The maximum size of the single data file or temp file is 128 terabytes (TB) for a tablespace with 32 K blocks and 32 TB for a tablespace with 8 K blocks.

Traditional smallfile tablespaces (which are the default) may contain multiple data files, but the files cannot be as large. For more information about bigfile tablespaces, see *Oracle Database Administrator's Guide*.

Default Tablespaces

Tablespace	Description
SYSTEM	The SYSTEM tablespace is used for core functionality.
SYSAUX	The SYSAUX tablespace is an auxiliary tablespace to the SYSTEM tablespace and helps reduce the load on the SYSTEM tablespace.
TEMP	The TEMP tablespace contains schema objects only for a session's duration.
UNDO	The UNDO tablespace stores the data needed to roll back, or undo, changes to the database.
USERS	The USERS tablespace stores user objects and data.



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

SYSTEM: It stores the data dictionary (metadata that describes the objects in the database) and tables that contain administrative information about the database. All this information is contained in the `SYS` schema and can be accessed only by the `SYS` user or other administrative users with the required privilege.

SYSAUX: The SYSAUX tablespace is an auxiliary tablespace to the SYSTEM tablespace and helps reduce the load on the SYSTEM tablespace. Some components and products that have used the SYSTEM tablespace or their own tablespaces in earlier releases of Oracle Database now use the SYSAUX tablespace.

TEMP: The TEMP tablespace contains schema objects only for a session's duration. Objects in temporary tablespaces are stored in temp files. Your temporary tablespace is used when you execute a SQL statement that requires the creation of temporary segments (such as a large sort or the creation of an index). Just as each user is assigned a default tablespace for storing created data objects, each user is assigned a temporary tablespace.

UNDO: The UNDO tablespace stores the data needed to roll back, or undo, changes to the database.

USERS: The USERS tablespace stores user objects and data. If no default tablespace is specified when a user is created, then the USERS tablespace is the default tablespace for all objects created by that user. For the `SYS` and `SYSTEM` users, the default permanent tablespace is `SYSTEM`.

SYSTEM and SYSAUX Tablespaces

- The SYSTEM and SYSAUX tablespaces are mandatory tablespaces that are created at the time of database creation. They must be online.
- The SYSTEM tablespace is used for core functionality (for example, data dictionary tables).
- The auxiliary SYSAUX tablespace is used for additional database components.
- The SYSTEM and SYSAUX tablespaces should not be used for application data.



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Each Oracle database must contain a SYSTEM tablespace and a SYSAUX tablespace. They are automatically created when the database is created. The system default is to create a smallfile tablespace. You can also create bigfile tablespaces, which enable the Oracle database to manage ultralarge files.

A tablespace can be online (accessible) or offline (not accessible). The SYSTEM tablespace is always online when the database is open. It stores tables that support the core functionality of the database, such as the data dictionary tables.

The SYSAUX tablespace is an auxiliary tablespace to the SYSTEM tablespace. The SYSAUX tablespace stores many database components and must be online for the correct functioning of all database components. The SYSTEM and SYSAUX tablespaces are not recommended for storing an application's data. Additional tablespaces can be created for this purpose.

Note: The SYSAUX tablespace may be taken offline to perform tablespace recovery, whereas this is not possible for the SYSTEM tablespace. Neither of them may be made read-only.

Implementing Oracle Managed Files (OMF)

- Specify file operations in terms of database objects rather than file names.

Parameter	Description
DB_CREATE_FILE_DEST	Defines the location of the default file system directory for data files and temporary files
DB_CREATE_ONLINE_LOG_DEST_n	Defines the location for redo log files and control file creation
DB_RECOVERY_FILE_DEST	Gives the default location for the fast recovery area

- Example:

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST='/u01/app/oracle/oradata';
SQL> CREATE TABLESPACE tbs_1;
```



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Managed Files

Oracle Managed Files (OMF) eliminates the need for you to directly manage the operating system files in an Oracle database. You specify operations in terms of database objects rather than file names. The database internally uses standard file system interfaces to create and delete files as needed for the following database structures:

- Tablespaces
- Redo log files
- Control files
- Archived logs
- Block change tracking files
- Flashback logs
- RMAN backups

A database can have a mixture of Oracle-managed and Oracle-unmanaged files. The file system directory specified by either of these parameters must already exist; the database does not create it. The directory must also have permissions for the database to create the files in it.

The table in the slide lists three initialization parameters that are used with OMF. The example in the slide shows that after the DB_CREATE_FILE_DEST initialization parameter is set, you can omit the DATAFILE clause from the CREATE TABLESPACE statement. The data file is created in the location specified by DB_CREATE_FILE_DEST.

In Oracle Database Cloud Service, OMF is enabled by default.

Naming Formats

Oracle-managed files have a specific naming format. For example, on Linux- and UNIX-based systems, the following format is used:

```
<destination_prefix>/o1_mf_%t_%u_.dbf
```

Do not rename an Oracle-managed file. The database identifies an Oracle-managed file based on its name. If you rename the file, the database will no longer be able to recognize it as an Oracle-managed file and will not manage the file accordingly.

File Size

By default, Oracle-managed data files, including those for the `SYSTEM` and `SYSAUX` tablespaces, are 100 MB and auto-extensible.

Note: By default, Automatic Storage Management (ASM) uses OMF files, but if you specify an alias name for an ASM data file at tablespace creation time or when adding an ASM data file to an existing tablespace, then that file will not be OMF.

Oracle Container Database: Introduction

- *Pluggable database*: Is a set of database schemas that appears logically to users and applications as a separate database
- *Multitenant container database*: Has a database instance and database files at the physical level
- All pluggable databases share:
 - Background processes
 - Shared/process memory
 - Oracle metadata



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

A pluggable database (PDB) is a set of database schemas that appears logically to users and applications as a separate database. But at the physical level, the multitenant container database (CDB) has a database instance and database files, just as a noncontainer database does.

It is easy to plug non-CDBs into a CDB.

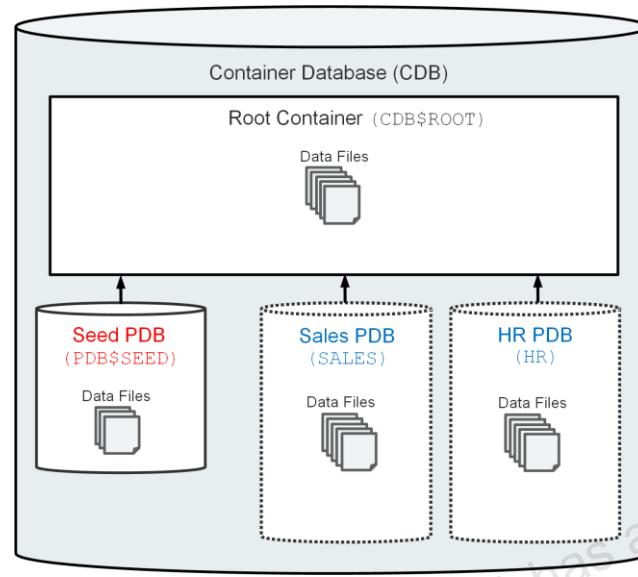
A CDB avoids redundancy of:

- Background processes
- Memory allocation
- Oracle metadata in several data dictionaries

A CDB grouping several applications has one instance, one set of background processes, one SGA allocation, and one data dictionary in the root container, common for all PDBs, each PDB maintaining its own application data dictionary.

When applications need to be patched or upgraded, the maintenance operation is performed only once on the CDB, and consequently, all applications are updated at the same time.

Multitenant Database



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

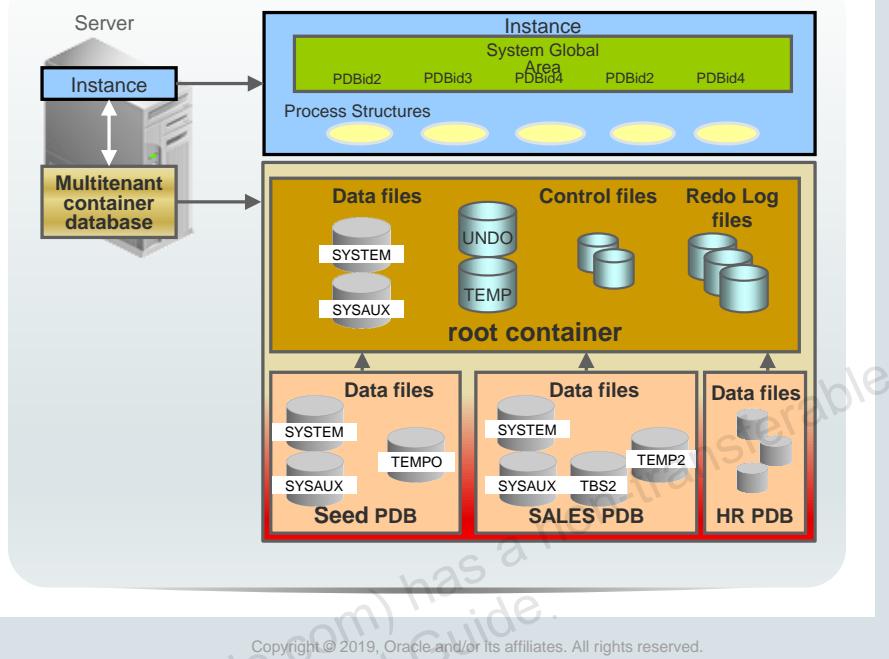
You can design your database to be a multitenant container database (CDB). A CDB, as illustrated in the slide, is made up of one root container, one seed pluggable database (seed PDB), and one or more user-created pluggable databases (simply referred to as PDBs). To a user or application, PDBs appear logically as separate databases. You can still configure the database the old way by using the noncontainer database (non-CDB) architecture, if needed.

- The root container, named `CDB$ROOT`, contains multiple data files. The data files store Oracle-supplied metadata and common users (users that are known in every container). This information is shared with all PDBs.
- The seed PDB, named `PDB$SEED`, is a system-supplied PDB template containing multiple data files that you can use to create new PDBs.
- The user-created PDB contains multiple data files that contain the data and code required to support an application (for example, a Human Resources application). Users interact only with the PDBs, and not the seed PDB or root container. For example, in the slide, there are two PDBs—one for the sales organization (named `SALES`) and another for the Human Resources department (named `HR`). You can create multiple PDBs in a CDB. One of the goals of the multitenant architecture is that each PDB has a one-to-one relationship with an application.

Multitenant Architecture

All PDBs share:

- Background processes
- Shared/process memory
- Oracle metadata
- Redo log files
- Control files
- Undo tablespace



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The graphic in the slide shows a CDB with four containers: the root, the seed, and two PDBs. The two applications (HR and SALES) use a single instance and are maintained separately.

At the physical level, the CDB has a database instance and database files, just as a non-CDB does.

- The redo log files are common for the whole CDB. The information it contains is annotated with the identity of the PDB where a change occurs. Oracle GoldenGate is enhanced to understand the format of the redo log for a CDB. All PDBs in a CDB share the ARCHIVELOG mode of the CDB.
- The control files are common for the whole CDB. The control files are updated to reflect any additional tablespace and data files of plugged PDBs.
- The UNDO tablespace is common for all containers.
- A temporary tablespace common to all containers is required. But each PDB can hold its own temporary tablespace for its own local users.
- Each container has its own data dictionary stored in its proper SYSTEM tablespace, containing its own metadata, and a SYSAUX tablespace.
- The PDBs can create tablespaces within the PDB according to application needs.
- Each data file is associated with a specific container, named *CON_ID*.

Default Tablespaces in the Multitenant Architecture

Tablespace	Description
SYSTEM	In the root container, it contains Oracle-supplied metadata. In a PDB, it contains user metadata.
SYSAUX	It exists in the root container and in each PDB.
TEMP	By default, the root container has a single default temporary tablespace that every PDB uses. You can create separate temporary tablespaces in PDBs.
UNDO	One active undo tablespace exists in the root container. It is recommended that you have a local undo tablespace in each PDB.
USERS	The root container and each PDB have a USERS tablespace.



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

SYSTEM: In the root container, it stores Oracle-supplied metadata, whereas in a PDB, it stores user metadata. Pointers from the PDBs to the Oracle-supplied objects allow the “system” objects to be accessed without duplicating them in the PDBs.

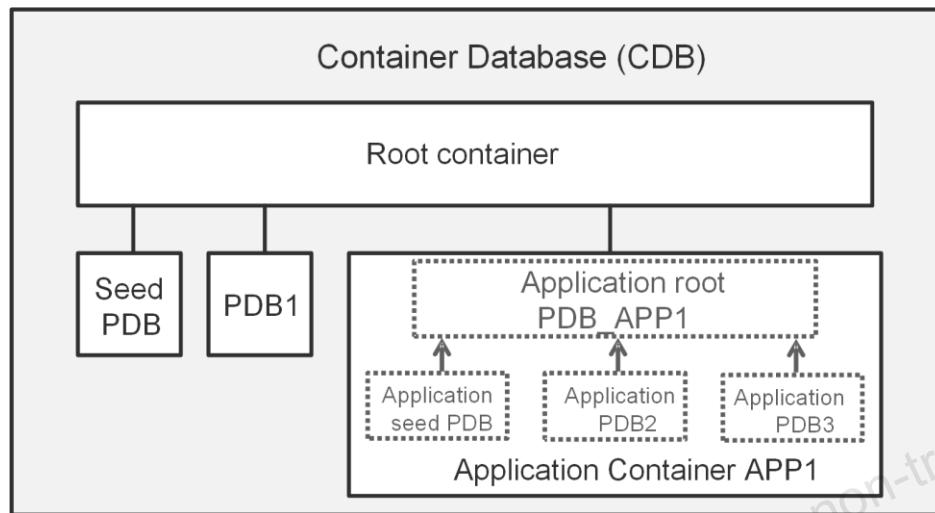
SYSAUX: The SYSAUX tablespace exists in the root container and in each PDB.

TEMP: By default, the root container has a single default temporary tablespace named TEMP that every PDB uses; however, if needed, you can create separate TEMP tablespaces in PDBs.

UNDO: In a single-instance CDB, one active UNDO tablespace exists in the root container. It is optional, but recommended, to have a local UNDO tablespace in a PDB.

USERS: The root container and each PDB have their own USERS tablespace.

Application Containers

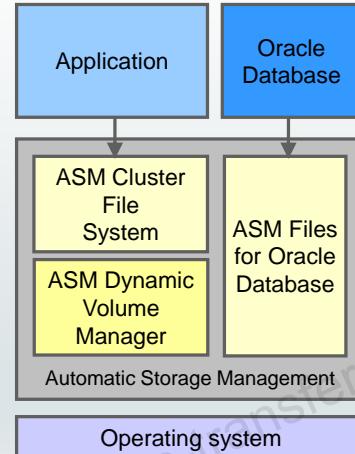


Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

An application container is a collection of PDBs within a CDB, storing data for an application. You create application containers to have a single master application definition. An application container, as shown in the slide, consists of an application root container, an optional application seed PDB, and application PDBs.

Automatic Storage Management

- Is a portable and high-performance cluster file system
- Manages Oracle database files
- Manages application files with ASM Cluster File System (ACFS)
- Spreads data across disks to balance load
- Mirrors data in case of failures
- Solves storage management challenges



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Automatic Storage Management (ASM) provides vertical integration of the file system and the volume manager for Oracle database files. ASM can provide management for single symmetric multiprocessing (SMP) machines or across multiple nodes of a cluster for Oracle Real Application Clusters (RAC) support.

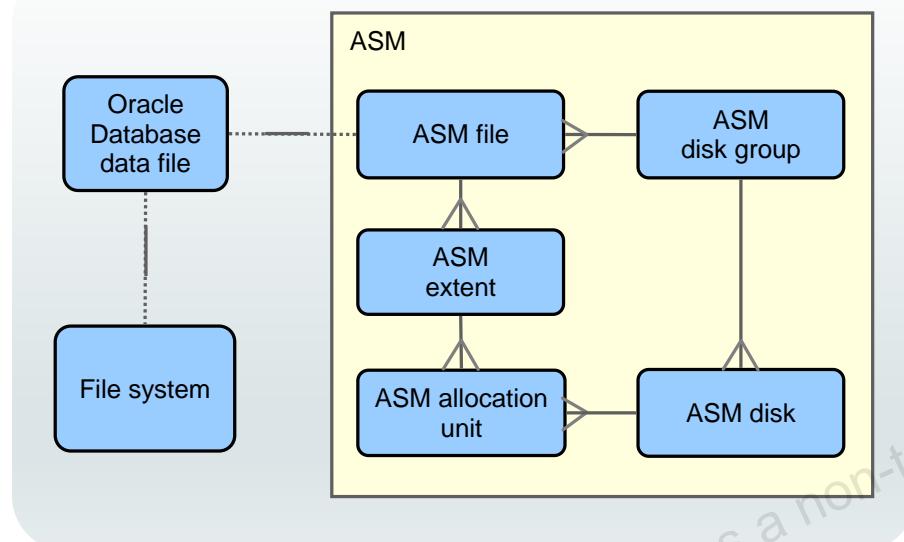
Oracle ASM Cluster File System (ACFS) is a multiplatform, scalable file system, and storage management technology that extends ASM functionality to support application files outside the Oracle Database, such as executables, reports, BFILEs, video, audio, text, images, and other general-purpose application file data.

ASM distributes input/output (I/O) load across all available resources to optimize performance while removing the need for manual I/O tuning. ASM helps database administrators (DBAs) manage a dynamic database environment by enabling them to increase the database size without having to shut down the database to adjust storage allocation.

ASM can maintain redundant copies of data to provide fault tolerance, or it can be built on top of vendor-supplied storage mechanisms. Data management is done by selecting the desired reliability and performance characteristics for classes of data rather than with human interaction on a per-file basis.

ASM capabilities save the DBA's time by automating manual storage and thereby increasing the administrator's ability to manage more and larger databases with increased efficiency.

ASM Storage Components



ORACLE®

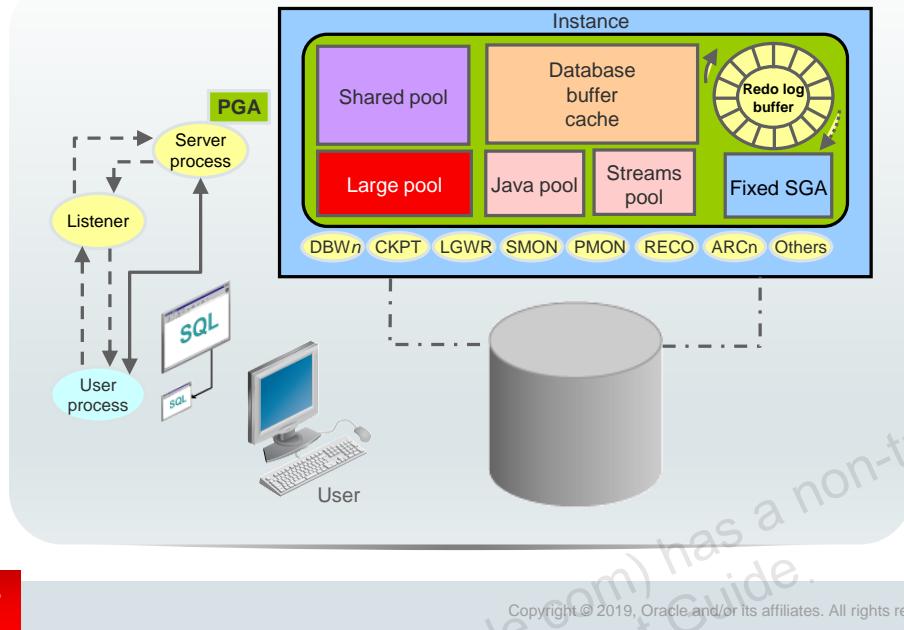
Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

ASM does not eliminate any existing database functionality. Existing databases are able to operate as they always have. New files may be created as ASM files, whereas existing ones are administered in the old way or can be migrated to ASM.

The diagram illustrates the relationships between an Oracle database data file and the ASM storage components. The crow's foot notation represents a one-to-many relationship. An Oracle Database data file has a one-to-one relationship with either a file stored on the operating system in a file system or an ASM file.

An Oracle ASM disk group is a collection of one or more Oracle ASM disks managed as a logical unit. The data structures in a disk group are self-contained using some of the space for metadata needs. Oracle ASM disks are the storage devices provisioned to an Oracle ASM disk group and can be a physical disk or partitions, a Logical Unit Number (LUN) from a storage array, a logical volume (LV), or a network-attached file. Each ASM disk is divided into many ASM allocation units, the smallest contiguous amount of disk space that ASM allocates. When you create an ASM disk group, you can set the ASM allocation unit size to 1, 2, 4, 8, 16, 32, or 64 MB, depending on the disk group compatibility level. One or more ASM allocation units form an ASM extent. An Oracle ASM extent is the raw storage used to hold the contents of an Oracle ASM file. An Oracle ASM file consists of one or more file extents. Variable extent sizes of 1*AU size, 4*AU size, and 16*AU size are used for supporting very large ASM files.

Interacting with an Oracle Database: Memory, Processes, and Storage



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The following example describes Oracle database operations at the most basic level. It illustrates an Oracle database configuration in which the user and associated server process are on separate computers, connected through a network.

1. An instance has started on a node where Oracle Database is installed, often called the *host* or *database server*.
2. A user starts an application spawning a user process. The application attempts to establish a connection to the server. (The connection may be local, client/server, or a three-tier connection from a middle tier.)
3. The server runs a listener that has the appropriate Oracle Net Services handler. The listener detects the connection request from the application and creates a dedicated server process on behalf of the user process.
4. The user runs a DML-type SQL statement and commits the transaction. For example, the user changes the address of a customer in a table and commits the change.
5. The server process receives the statement and checks the shared pool (an SGA component) for any shared SQL area that contains an identical SQL statement. If a shared SQL area is found, the server process checks the user's access privileges to the requested data, and the existing shared SQL area is used to process the statement. If a shared SQL area is not found, a new shared SQL area is allocated for the statement so that it can be parsed and processed.
6. The server process retrieves any necessary data values, either from the actual data file (table) or from values stored in the database buffer cache.
7. The server process modifies data in the SGA. Because the transaction is committed, the Log Writer process (LGWR) immediately records the transaction in the redo log file. The Database Writer process (DBWn) writes modified blocks permanently to disk when it is efficient to do so.

8. If the transaction is successful, the server process sends a message across the network to the application. If it is not successful, an error message is transmitted.
9. Throughout this entire procedure, the other background processes run, watching for conditions that require intervention. In addition, the database server manages other users' transactions and prevents contention between transactions that request the same data.

Summary

In this lesson, you should have learned how to:

- List the major architectural components of Oracle Database
- Explain memory structures
- Describe background processes
- Correlate logical and physical storage structures
- Describe multitenant architecture



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

3

Introduction to Oracle Database Cloud Service



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Explain the Oracle Cloud and Database Cloud offerings
- Describe the architecture and features of Oracle Database Cloud Service



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Cloud: Overview

- **Oracle Cloud Software as a Service (SaaS)**
 - Provides best-of-breed cloud software in a complete, secure, and connected cloud suite
 - Comes embedded with modern best practice processes and built-in social, mobile, and analytic capabilities
- **Oracle Cloud Platform as a Service (PaaS)**
 - Enables enterprise IT and independent software vendor (ISV) developers to rapidly build and deploy rich applications
 - Uses an enterprise-grade cloud platform based on the industry's best database and application server
- **Oracle Cloud Infrastructure as a Service (IaaS)**
 - Is a set of core infrastructure capabilities such as elastic compute and storage
 - Provides customers with the ability to run any workload in the cloud



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Cloud is the industry's broadest and most integrated public cloud. It offers best-in-class services across software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS).

For additional information, see the Oracle Cloud website at <https://cloud.oracle.com>.

Database Cloud Service Offerings

- Oracle Database Cloud Service
 - Offers elastic database services for application development, testing, and production deployment
 - Has an easy-to-use web console user interface and RESTful API to provision and administer Oracle Database on Oracle Compute Cloud
- Oracle Database Exadata Cloud Service
 - Includes all the benefits of Exadata performance
 - Offers 100% compatibility with existing business-critical applications and databases that are on premises
- Oracle Database Exadata Express Cloud Service—Managed
 - Provides a full Oracle Database experience for small- and medium-sized databases
 - Has service that is fully managed by Oracle



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Database Cloud Service: Oracle offers general-purpose and high-memory compute shapes to provide the full power of the Oracle Database in the cloud for any type of application. You can use all standard network connections and have administrative control.

Oracle Database Exadata Cloud Service: Oracle Database Exadata Cloud Service delivers the world's best cloud database platform by combining the world's #1 database with Exadata, the most powerful database platform, and adding all the simplicity and cost effectiveness of the public cloud.

Oracle Database Exadata Express Cloud Service: Exadata Express provides your own Oracle Database Enterprise Edition running the latest database release on Exadata for a full Oracle experience. It's a fully managed service packed with features for modern application development and great for small- to medium-sized data.

Infrastructure for Oracle Database Cloud Service

- Oracle Cloud Infrastructure Classic (OCI Classic)
 - Oracle's first-generation cloud infrastructure
 - Includes Compute Classic and Storage Classic
- Oracle Cloud Infrastructure (OCI)
 - Oracle's second-generation cloud infrastructure
 - Includes Compute and Storage



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can create Oracle Database Cloud Service database deployments in Oracle Cloud Infrastructure Classic (OCI Classic) and in Oracle Cloud Infrastructure (OCI).

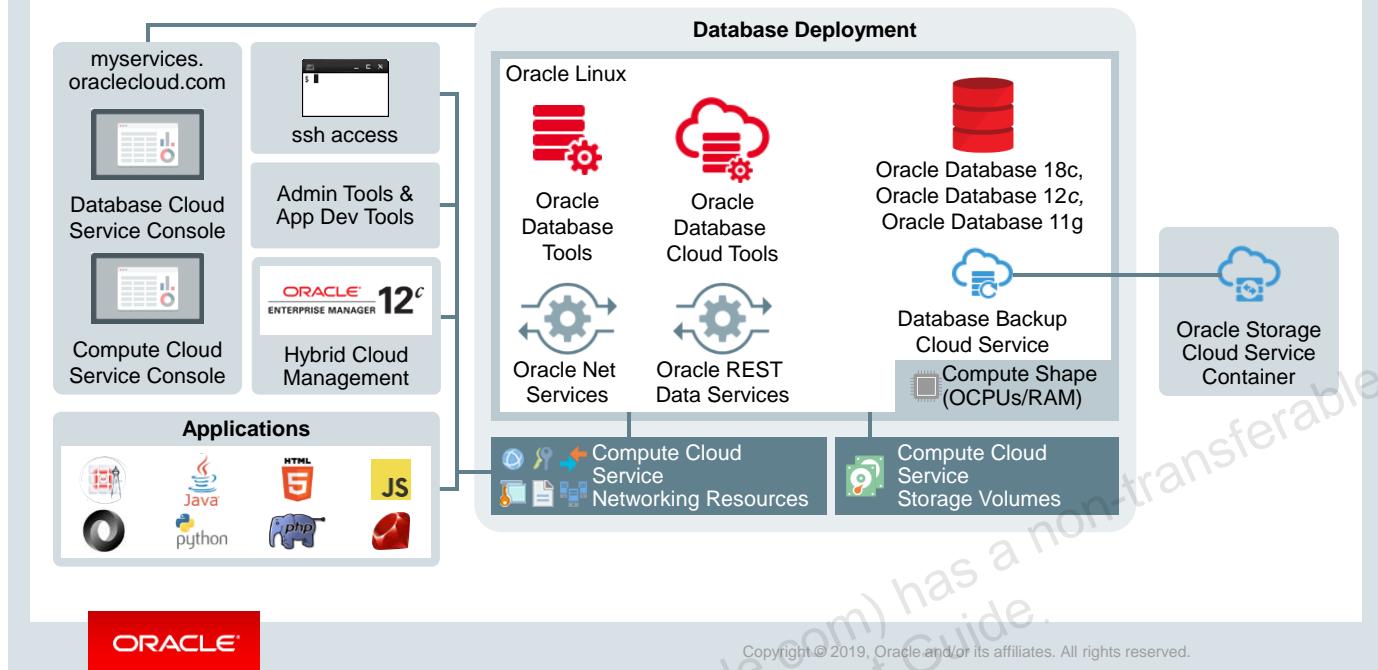
OCI Classic is an integrated infrastructure service that enables businesses to optimally run Oracle workloads in an enterprise cloud managed, hosted, and supported by Oracle. See <https://cloud.oracle.com/classic> for detailed information on OCI Classic.

OCI, Oracle's second-generation cloud infrastructure, combines the elasticity and utility of public cloud with the granular control, security, and predictability of on-premises infrastructure to deliver high-performance, high-availability and cost-effective infrastructure services. See <https://cloud.oracle.com/cloud-infrastructure> for detailed information on OCI.

The Oracle Database environment in either type of infrastructure is substantially the same. There are a few differences in the underlying infrastructure components and in the supported capabilities. Awareness of these differences will help you choose an appropriate infrastructure when creating a database deployment.

See "About Database Deployments in Oracle Cloud Infrastructure" in *Administering Oracle Database Cloud Service* for additional information.

Database Cloud Service Architecture (OCI Classic)



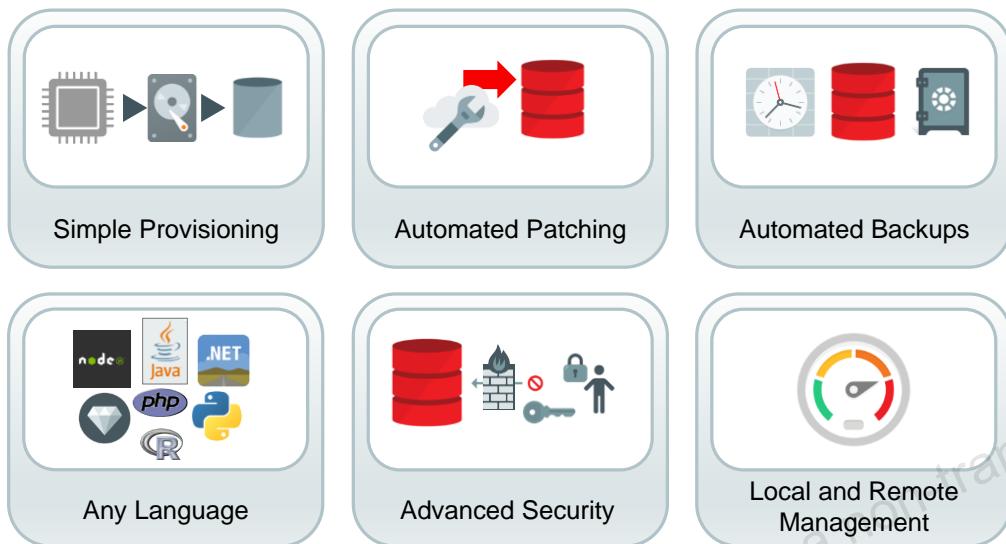
Oracle Database Cloud Service provides you the ability to deploy Oracle databases in the cloud, with each database deployment containing a single Oracle database. You have full access to the features and operations available with Oracle Database, but with Oracle providing the computing power, physical storage, and (optionally) tooling to simplify routine database maintenance and management operations.

When you create database deployments, Database Cloud Service creates compute nodes to host the database by using computing and storage resources provided by Oracle Compute Cloud Service. Additionally, it provides access to the compute nodes (and thus to the database) by using networking resources provided by Oracle Compute Cloud Service.

Oracle Database Cloud Service includes Oracle Database and supporting software. An Oracle database is created by using values you provide when creating the database deployment, and the database is started. Additionally, you can direct Database Cloud Service to set up automatic backups. Finally, the deployment includes cloud tooling that simplifies backup, recovery, patching, and upgrade operations.

Note: In this course, the OCI Classic infrastructure is used.

Features and Tooling



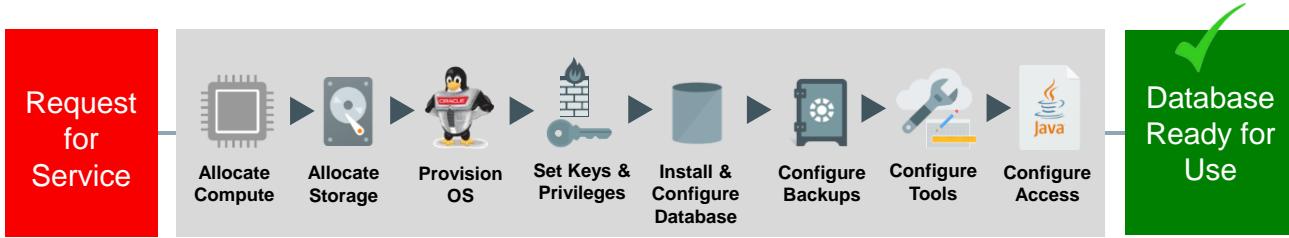
ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Database Cloud Service includes the following features and tools:

- **Simple Provisioning:** The Create Database Cloud Service Instance wizard is used to create a new database deployment.
- **Automated Patching:** Available patches are displayed through the Database Cloud Service console and can be applied with a single click.
- **Automated Backups:** Automatic backups can be set up. Cloud tooling provides for simplified backup and recovery operations.
- **Any Language:** Database Cloud Service supports any language.
- **Advanced Security:** Comprehensive security, including encryption of data at rest and data in transit.
- **Local and Remote Management:** Enterprise Manager Database Control for Oracle Database 11g databases and Enterprise Manager Database Express for Oracle Database 12c and 18c databases are part of the installation and can be used to monitor and manage the database. Enterprise Manager Cloud Control 13c can also be used to manage the database.

Automated Database Provisioning



Automated provisioning based on input to the Create Oracle Database Cloud Service Instance wizard or through REST APIs

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

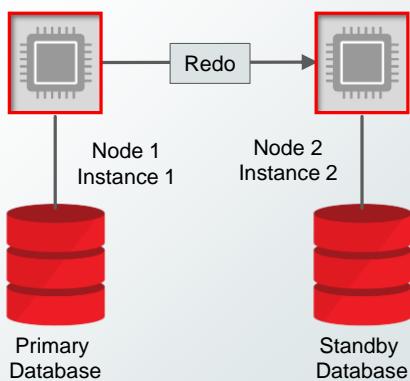
The Create Oracle Database Cloud Service Instance wizard is used to create a new database deployment.

After requesting the creation of the database deployment, the steps shown in the gray box happen automatically.

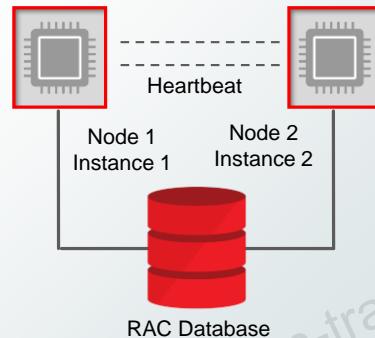
When the creation and configuration is complete, the database deployment is ready for use.

Additional Database Configuration Options

- Oracle Data Guard



- Oracle Real Application Clusters (Oracle RAC)



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Data Guard

You can optionally create an Oracle Data Guard configuration when you create a database deployment.

Oracle Data Guard enables production Oracle databases to survive disasters and data corruptions by providing a comprehensive set of services that create, maintain, manage, and monitor a standby database. Oracle Data Guard maintains the standby database as a copy of the production database. If the production database becomes unavailable because of a planned or an unplanned outage, you can switch the standby database to the production role, minimizing the down time associated with the outage.

See *Using Oracle Data Guard in Database as a Service* for additional information.

Oracle Real Application Clusters

You can optionally create an Oracle Real Application Clusters (RAC) database when you create a database deployment.

Oracle RAC enhances Oracle Database capabilities so that you can concurrently use multiple database instances on different compute nodes. This allows you to scale workload across multiple database instances to efficiently store, update, and retrieve data.

Oracle RAC provides the software that manages multiple servers and database instances as a single set of servers, called a cluster. The data files that comprise the database reside on shared storage that is accessible from all servers that are part of the cluster. Each server in the cluster runs the Oracle RAC software.

See *Using Oracle Real Application Clusters (RAC) in Database as a Service* for additional information.

Summary

In this lesson, you should have learned how to:

- Explain the Oracle Cloud and Database Cloud offerings
- Describe the architecture and features of Oracle Database Cloud Service



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 3: Overview

- 3-1: Exploring Oracle Cloud



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

4

Creating DBCS Database Deployments

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

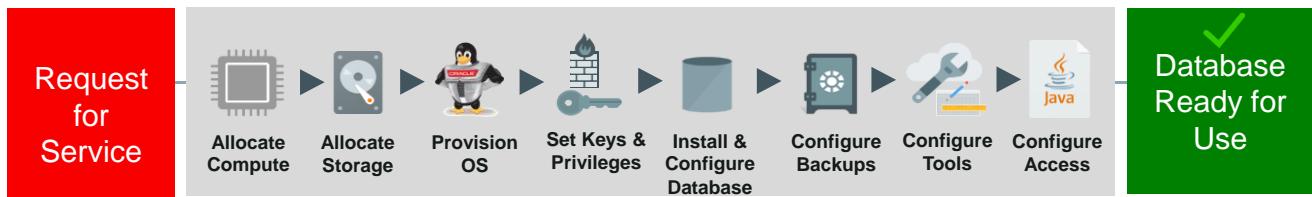
- Create a DBCS database deployment
- Describe how an SSH key pair is used in authentication
- Explain how storage volumes are allocated for a DBCS database deployment



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Automated Database Provisioning



Automated provisioning based on input to the Create Oracle Database Cloud Service Instance wizard or through REST APIs

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Create Oracle Database Cloud Service Instance wizard is used to create a new database deployment. After requesting the creation of the database deployment, the steps shown in the gray box happen automatically. When the creation and configuration are complete, the database deployment is ready for use.

Creating a Database Deployment

The wizard takes you through the database deployment creation, prompting for the following information:

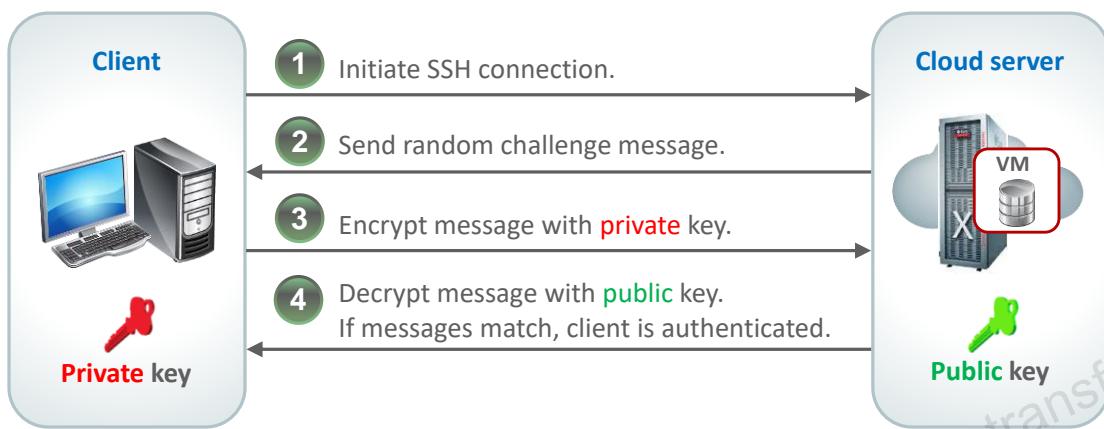
- Subscription: Service level, billing frequency, software release, and software edition
- Service: Service name, compute shape, time zone, and SSH key
- Database Configuration: Storage, database name (SID), PDB name, character set, and password for administrative accounts
- Backup and Recovery Configuration: Backup destination and cloud storage information, if applicable
- Additional Options: Real Application Clusters (RAC), Data Guard, and GoldenGate



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

See “Creating a Database Deployment” in *Administering Oracle Database Cloud Service* for the most current information about using the wizard.

How SSH Key Pairs Are Used



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

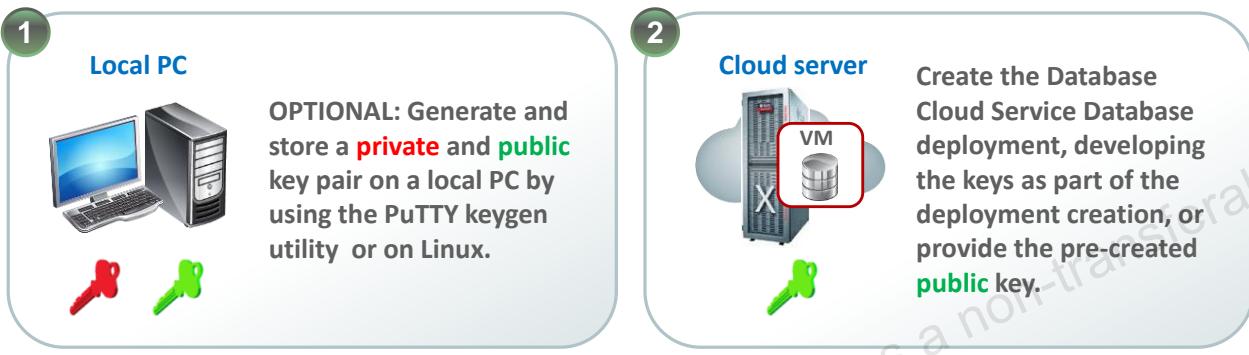
By default, network access to the compute nodes associated with Oracle Database Cloud Service is provided by Secure Shell (SSH) connections on port 22. SSH is a cryptographic network protocol that uses two keys, one public and one private, to provide secure communication between two networked computers. Port 22 is the standard TCP/IP port that is assigned to SSH servers.

See “About Network Access to Database Cloud Service” in *Administering Oracle Database Cloud Service* for additional information.

Creating an SSH Key Pair

Perform one of the following:

- Generate an SSH key pair as part of the database deployment creation.
- Create an SSH key pair before creating your database deployment.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When you create a database deployment, you can create an SSH key pair through the wizard or you can upload a public key file that you have created by using a utility.

See “Generating a Secure Shell (SSH) Public/Private Key Pair” in *Administering Oracle Database Cloud Service* for additional information.

Storage Used for Database Files

Storage Volume	Description
<code>bits</code>	30 GB volume completely allocated to <code>/u01</code> on the compute node.
<code>boot</code>	21 GB volume allocated to the following file system mounts on the virtual machine: <code>/</code> (root), <code>/boot</code> , and swap space.
<code>data</code>	GB size equal to the value provided in the Usable Data Storage field during the database deployment creation process, with a minimum of 11 GB. This volume is completely allocated to <code>/u02</code> on the compute node.
<code>fra</code>	If backups are configured, GB size equal to 1.7 times the size of the <code>data</code> volume. If backups are not configured, GB size equal to 0.1 times the size of the <code>data</code> volume, with a minimum of 7 GB. This volume is completely allocated to <code>/u03</code> on the compute node.
<code>redo</code>	10 GB volume completely allocated to <code>/u04</code> on the compute node.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When a database deployment is created, Oracle Cloud Service storage volumes are created and allocated to the database deployment's compute node.

See “Storage Volumes and File System Layout” in *Administering Oracle Database Cloud Service* for additional information.

Summary

In this lesson, you should have learned how to:

- Create a DBCS database deployment
- Describe how an SSH key pair is used in authentication
- Explain how storage volumes are allocated for a DBCS database deployment



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 4: Overview

- 4-1: Creating a Database Deployment



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

5

Accessing an Oracle Database



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Connect to an Oracle Database
- Describe the tools used to access an Oracle Database
- Describe the Oracle-supplied user accounts for Oracle Database
- Query the data dictionary in an Oracle Database



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Connecting to an Oracle Database Instance

- You connect client applications to an Oracle Database by connecting to its database instance, not its database.
- A user session is a logical entity that represents the state of the current user login to the database instance.
- Examples of connecting to an Oracle Database instance:
 - By using operating system authentication

```
$ sqlplus / as sysdba
```

- By using Easy Connect Syntax

```
SQL> connect hr/hr@host1.example.com:1521/db.example.com
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Connections Compared With Sessions

You connect client applications to database instances (not databases).

A connection is the physical communication pathway between a client process and a database instance.

A user session is a logical entity that represents the state of the current user login to the database instance. A session lasts from the time the user is authenticated by the database instance until the time the user disconnects or exits the client application.

Connecting to CDBs by Using Operating System Authentication

As a database administrator, you can quickly start SQL*Plus and connect to a root container without a password by using the `$ sqlplus / as sysdba` command. This command enables you to connect to the database as the `SYS` user with the `SYSDBA` privilege. There are some rules: You must be on the same machine as the database instance, and the current operating system user must be a member of the privileged `OSDBA` group.

Connecting to PDBs by Using the Easy Connect Syntax

There are many ways to connect to a PDB; however, using the Easy Connect syntax in SQL*Plus is the easiest because it's already enabled on the database server by default and doesn't require any client-side configuration. This syntax supports TCP protocol only (no SSL). It offers no support for advanced connection options such as connect-time failover, source routing, and load balancing.

Easy Connect connection strings take the following form:

```
SQL> CONNECT <username>/<password>@<listener hostname>:<listener port>/<service name>
```

For example, the SYSTEM user requests a connection to the database service named db.example.com. The listener is located on a machine named host01.example.com and listens on port 1521.

```
SQL> CONNECT hr/hr@host1.example.com:1521/db.example.com
```

If you're starting from a command prompt, you can start SQL*Plus and log in at the same time:

```
$ sqlplus hr/hr@host1.example.com:1521/db.example.com
```

The listener port and service name are optional. If the listener port is not provided, Oracle Net assumes that the default port of 1521 is being used. If the service name is not provided, Oracle Net assumes that the database service name and host name provided in the connect string are identical. For example, assuming that the listener uses TCP to listen on port 1521, the connection string above can be shortened.

For example, a connection string like this:

```
SQL> CONNECT hr/hr@db.example.com:1521/db.example.com
```

...can be shortened to:

```
SQL> CONNECT hr/hr@ db.example.com
```

Disconnecting from the Database Instance

Use the EXIT command to exit SQL*Plus, disconnect from the database instance, and end all sessions in the database instance memory.

Oracle Database Tools

- Oracle Database tools each have their own purpose, and some operations can be performed in more than one tool.
- Finding the right tool for the job often comes down to preference (for example, whether you prefer to work with code or use a graphical user interface).
- Tools include:
 - SQL*Plus
 - SQL Developer
 - SQL Developer Command Line (SQLcl)
 - Database Configuration Assistant (DBCA)
 - Oracle Enterprise Manager Database Express (EM Express)
 - Oracle Enterprise Manager Cloud Control
 - Others such as Listener Control, Oracle Net Configuration Assistant, Oracle Net Manager, ADR Command Interpreter, SQL*Loader, Oracle Data Pump Import, and Oracle Data Pump Export



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Database includes the following tools:

- **SQL*Plus:** Use this command-line tool to access a database.
- **SQL Developer:** Use this graphical user interface (GUI) tool to access a database and perform DBA actions.
- **SQL Developer Command Line (SQLcl):** Use this tool to access a database.
- **Database Configuration Assistant (DBCA):** Use this GUI tool to create databases. To use DBCA, you must be on the server to launch the tool from the operating system that houses the CDB.
- **Oracle Enterprise Manager Database Express (EM Express):** Use this GUI tool to perform database administration tasks on one database instance at a time.
- **Oracle Enterprise Manager Cloud Control (EM Cloud Control):** Use this GUI tool to perform database administration tasks on several targets (database instances, listeners, and so on) at the same time.
- **Others:** Many specialized utilities are used to assist with database administration. The ones used in this course may include Listener Control (`lsnrctl`), Oracle Net Configuration Assistant (`netca`), Oracle Net Manager (`netmgr`), ADR Command Interpreter (`adcri`), SQL*Loader (`sqlldr`), Oracle Data Pump Import (`impdp`), and Oracle Data Pump Export (`expdp`). This list does not cover all the utilities available.

Database Tool Choices

Topic	SQL*Plus	SQL Developer	SQLcl	DBCA	EM Database Express	EM Cloud Control	Oracle Universal Installer
Create a CDB or PDB	Yes	Yes (PDB only)	Yes	Yes	Yes (PDB only)	Yes (PDB only)	Yes
Explore CDB instance, architecture, and PDBs	Yes	Yes	Yes	No	Yes	Yes	No



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The table shows you at a glance which tool can be used to perform which tasks.

SQL*Plus

- Example 1: From a command line, you can start SQL*Plus, log in, and show the user that you're logged in as:

```
$ sqlplus / as sysdba
...
SQL> show user
USER is "SYS"
```

- Example 2: Call a SQL script from the command line:

```
$ sqlplus hr/hr@HRPDB @script.sql
```

Username and password

Connect identifier

File name



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

SQL*Plus

SQL*Plus is a command-line program that you use to submit SQL and PL/SQL statements to an Oracle database. SQL*Plus is installed with Oracle Database and is located in the \$ORACLE_HOME/bin directory. You can start SQL*Plus from the command line or from the Start menu on a Windows client. Use the SQL*Plus command-line interface to execute SQL*Plus, SQL, and PL/SQL commands to perform the following:

- Enter, edit, run, store, retrieve, and save SQL commands and PL/SQL blocks
- Format, calculate, store, and print query results
- List column definitions for any table
- Send messages to and accept responses from an end user
- Perform database administration

Calling a SQL Script from SQL*Plus

When calling a SQL script file from within SQL*Plus, you have the following options:

- Option 1: Call the script from the command line when you first invoke SQL*Plus:
`$ sqlplus hr/hr@HRPDB @script.sql`
- Option 2: Call the script from inside a SQL*Plus session simply by using the "@" operator:
`SQL> @script.sql`

When a script is saved from SQL*Plus by using the `SAVE` command, the `.sql` extension is automatically supplied. You can then execute the script without supplying the extension at execution time, for example:

```
SQL> @script
```

Calling SQL*Plus from a Shell Script

You can call SQL*Plus from a shell script or BAT file by invoking `sqlplus` and using the operating system scripting syntax for passing parameters. For example, suppose you have the following shell script:

```
# Name of this file: batch_sqlplus.sh
# Count employees and give raise.
sqlplus hr/hr <<EOF
select count(*) from employees;
update employees set salary = salary*1.10;
commit;
quit
EOF
```

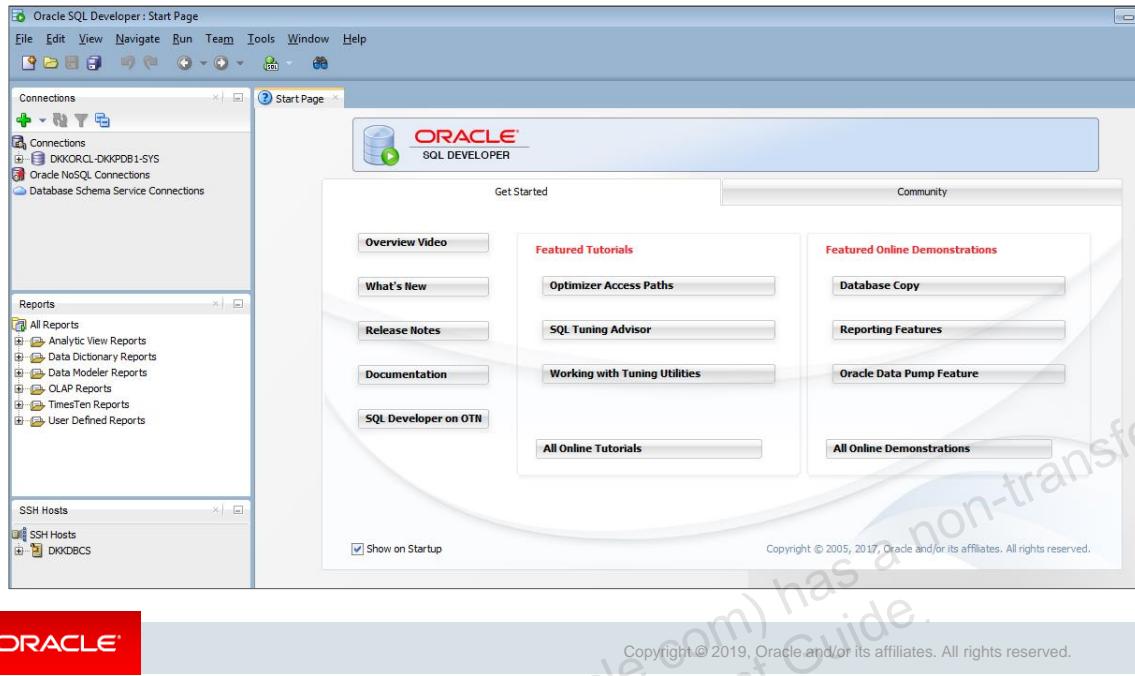
You can call that shell script from the command line:

```
$ ./batch_sqlplus.sh
```

For More Information

To learn how to start SQL*Plus, see “Starting SQL Plus” in *SQL*Plus Users Guide and Reference*.

Oracle SQL Developer



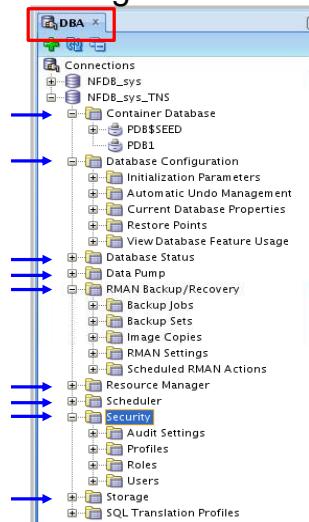
Oracle SQL Developer (SQL Developer) is a graphical tool for database developers and DBAs and gets installed with Oracle Database. You can choose to display several different panes in the SQL Developer interface, such as a Connections and DBA pane. You use the former to define connections to databases and use the latter to perform DBA operations. You should add connections only for which the associated database user has DBA privileges or at least privileges for the desired DBA navigator operations on the specified database.

Some developer-type operations that you can perform with SQL Developer include:

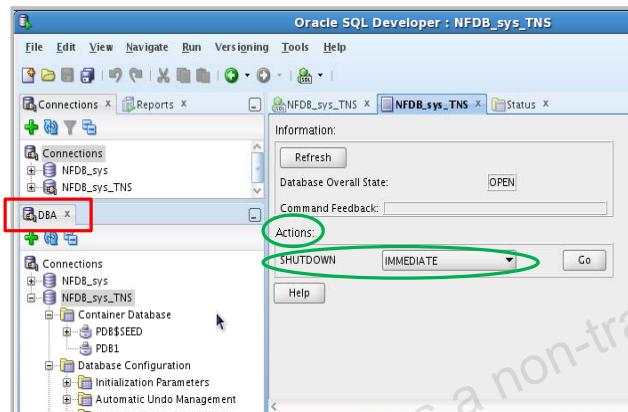
- Develop scripts in both the SQL and PL/SQL languages
- Browse database objects
- Run SQL statements and SQL scripts
- Edit and debug PL/SQL statements

Oracle SQL Developer: DBA Actions

Using DBA features through DBA navigator



Performing DBA actions



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Some DBA-type operations that you can perform with SQL Developer include:

- Start and shut down PDBs
- Configure databases (for example, configure initialization parameters)
- View the status of databases
- Export databases by using Data Pump and import jobs
- Back up and recover data by using Oracle Recovery Manager (RMAN)
- Configure Oracle Database Resource Manager (the Resource Manager), which enables you to manage multiple workloads within a database that are contending for system and database resources
- Schedule jobs (for example, loading data)
- Configure security by using audit settings, profiles, roles, and users
- Configure storage for archive logs, control files, data files, redo log groups, tablespaces, and temporary tablespace groups
- Run any number of provided reports, as well as create new ones

SQL Developer Command Line (SQLcl)

- SQLcl:
 - Is a tool that can access Oracle databases
 - Is a cross between SQL*Plus and SQL Developer
 - Incorporates Linux-like and IDE features, and color
- Example query with formatting to fit the screen:

```
SQL> SELECT first_name, last_name, salary, hire_date FROM hr.employees;  
  
FIRST_NAME      LAST_NAME      SALARY      HIRE_DATE  
Steven          King           24000       17-JUN-03  
Neena           Kochhar        17000       21-SEP-05  
...  
...
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

SQL Command Line (SQLcl) is another tool that you can use to access an Oracle Database. SQLcl is a cross between SQL Developer and SQL*Plus. It can do everything that SQL*Plus can do, plus more. It has improvements in command-line interaction, incorporating Linux-like and IDE features, and it includes color.

You can download SQLcl from the Oracle SQL Developer product page on the Oracle Technology Network at <http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html>

Note: For SQLcl to run on Windows or Linux operating systems, you need a Java Runtime Environment (JRE) of version 1.7 or higher.

See *Oracle SQL Developer Command Line Quick Reference* for details.

Database Configuration Assistant (DBCA)

- DBCA is a tool for creating and configuring an Oracle database
- DBCA has two modes:
 - Interactive: Provides a graphical interface and guided workflow
 - Noninteractive/silent: Uses command-line arguments, a response file, or both
- DBCA can be launched by the Oracle Universal Installer or invoked after the software is installed.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

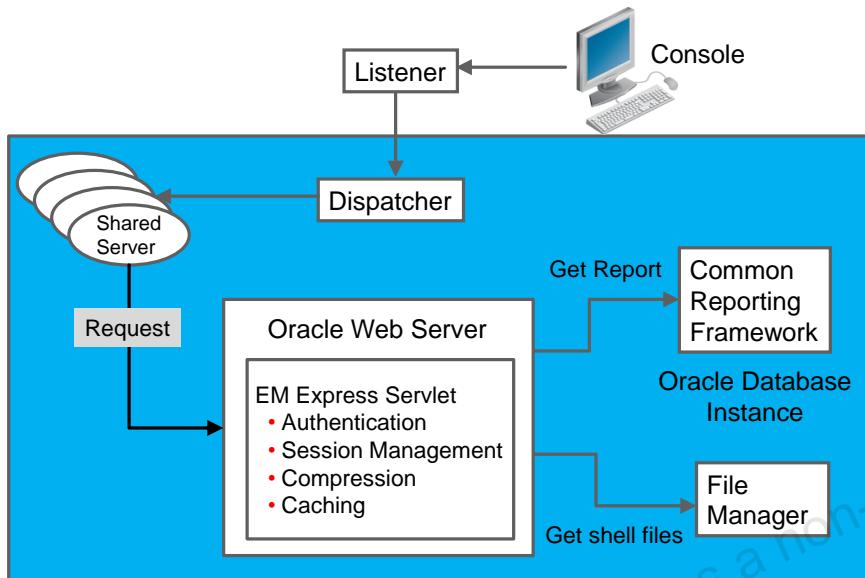
Database Configuration Assistant (DBCA) is a tool that you can use to do the following:

- Create databases (CDBs and non-CDBs)
- Configure existing databases
- Delete databases
- Manage templates
- Manage pluggable databases
- Manage Oracle RAC database instances

DBCA can be launched by the Oracle Universal Installer (OUI), depending upon the type of install that you select. You can also launch DBCA as a standalone tool at any time after Oracle Database installation.

See *Oracle Database Administrator's Guide* and *Oracle Database 2 Day DBA* for detailed information.

Oracle Enterprise Manager Database Express



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Enterprise Manager Database Express

Oracle Enterprise Manager Database Express (EM Express) is a lightweight tool that you can use to manage a CDB and all its PDBs (except the seed PDB). It provides an out-of-the-box browser-based management solution, performance monitoring, configuration management, administration, diagnostics, and tuning.

Architecture

EM Express uses a web-based console, communicating with the built-in web server available in XML DB.

As requests from the console are processed, the EM Express servlet handles the requests, including authentication, session management, compression, and caching. The servlet passes requests for reports to the Common Reporting Framework and actions requiring shell files to the File Manager. This architecture is illustrated in the slide.

EM Express is available only when the database is open. This means that Enterprise Manager Database Express cannot be used to start the database. Other operations that require that the database change state, such as enable or disable ARCHIVELOG mode, are also not available in EM Express.

EM Express is configurable with a single click in Database Configuration Assistant (DBCA).

EM Express is a servlet built on top of Oracle XML DB. The Oracle XML DB default wallet has a self-signed certificate, and some existing browsers consider self-signed certificates as untrusted because they are not signed by a trusted CA (certificate authority). However, the self-signed certificate is still secure, as it ensures that the traffic is encrypted between the Oracle XML DB server and the client (browser). Therefore, enter a security exception for the EM Express URL in your web browser.

Requirements

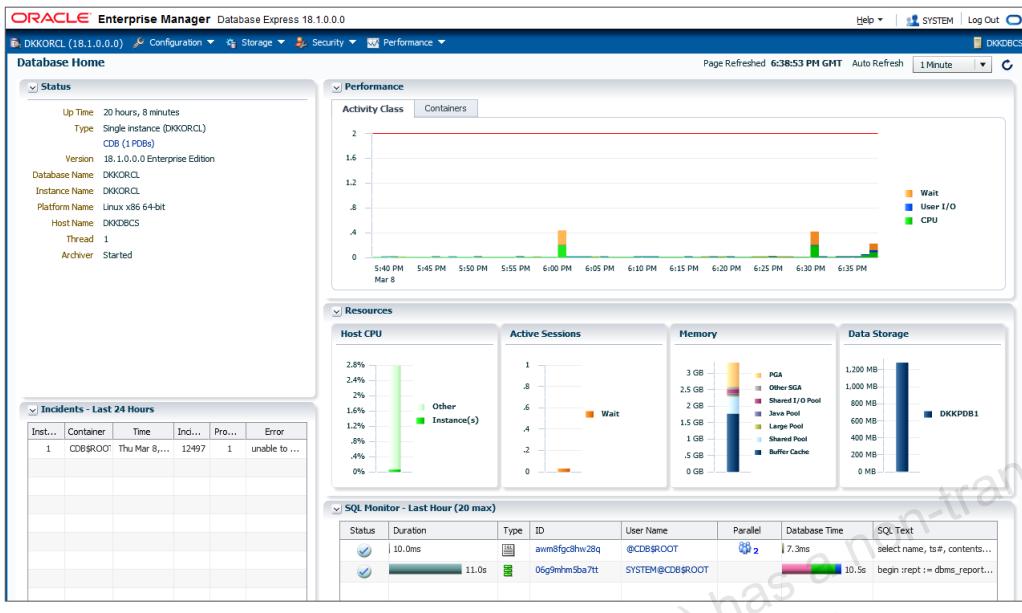
The following are required for EM Express:

- XMLDB components must be installed on the Oracle Database server. All Oracle Databases of version 12.1.0 or higher have XMLDB installed.
- The web browser must have the Flash plug-in installed because EM Express uses Shockwave Flash (SWF) files.

Starting EM Express for CDBs and PDBs

EM Express uses a global HTTPS port to connect to and manage non-CDBs, CDBs, and PDBs. With Oracle Database 12c Release 1, Enterprise Manager Database Express uses a different port for each PDB. Starting with Oracle Database 12c Release 2, you can set a global port that enables Enterprise Manager Database Express access to the CDB and all PDBs on that single port.

Using the Database Home Page

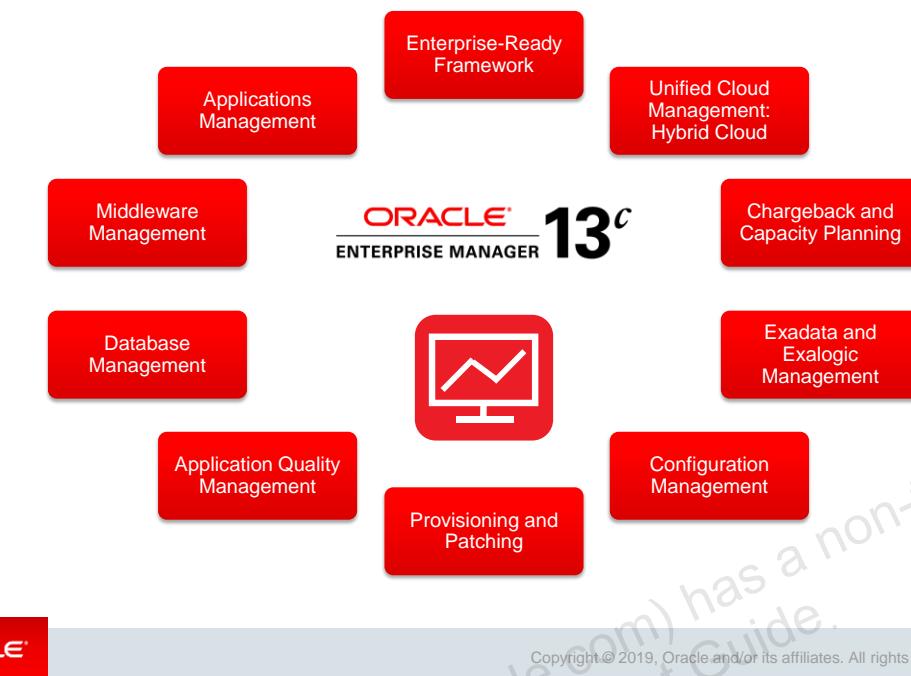


ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Enterprise Manager Database Express home page presents an overall view of the database instance status and activity.

Enterprise Manager Cloud Control 13c Features



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Enterprise Manager Cloud Control (EM Cloud Control) is Oracle's on-premises management platform, providing a single location for managing all your Oracle deployments, whether they be in your data centers or in Oracle Cloud. Through deep integration with Oracle's product stack, EM Cloud Control provides management and automation support for Oracle applications, databases, middleware, hardware, and engineered systems.

Key objectives in the design of Enterprise Manager Cloud Control 13c include:

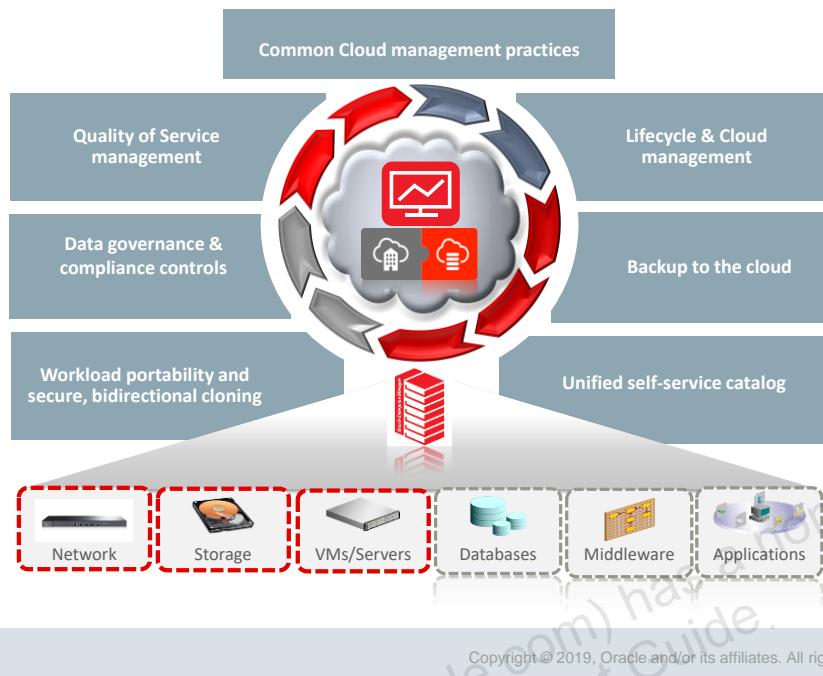
- Providing a complete integrated cloud management solution for a combination of on-premises cloud configurations and Oracle Cloud Services solutions (Hybrid Cloud)
- Delivering enhanced engineered systems management
- Enhancing middleware and database management
- Maintaining a robust, cloud-scale platform

Enterprise Manager Cloud Control 13c includes the following features:

- **Enterprise-Ready Framework:** Provides modular and extensible architecture, self-updateable entities, centralized incident management, in-context diagnostics management, as well as flexible job scheduling and security sub-systems
- **Cloud Management:** Provides complete cloud life cycle management for both on-premises clouds and PaaS services on Oracle Cloud
- **Chargeback and Capacity Planning:** Provides chargeback based on target types and uses Automatic Workload Repository (AWR) Warehouse to consolidate AWR reports from multiple databases across the enterprise

- **Exadata and Exalogic Management:** Provides an integrated view of the hardware and software in an Exadata machine and complete life cycle management for Exalogic systems
- **Configuration and Management:** Provides an integrated set of tools, agent-less discovery, integration with My Oracle Support, and custom configuration capabilities
- **Provisioning and Patching:** Provides profiles for provisioning known configurations, user-defined deployment procedures, and a Software Library integrated with self-updating capabilities
- **Application and Quality Management:** Provides Database Replay, Application Server Replay, Real Application Testing integrated with Data Masking, and test database management that includes Application Data Model
- **Database Management:** Provides management of Oracle Database systems, including performance management and change life cycle management
- **Middleware Management:** Provides complete management of Middleware systems
- **Applications Management:** Provides management of Fusion Applications

Single Pane of Glass for Enterprise Management



One Management Tool to Oversee Them All

Enterprise Manager Cloud Control 13c has capabilities to intelligently manage traditional and cloud-based services, mitigating the need to use multiple management and monitoring tools for what were previously two disparate environments.

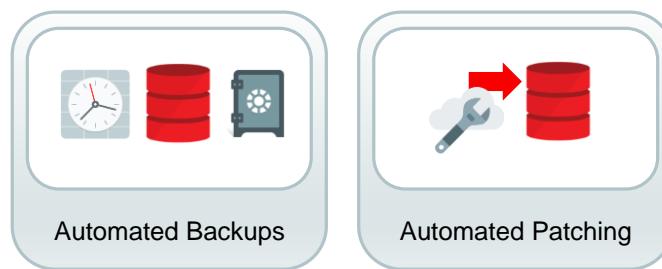
- Complete solution for management of the Oracle stack, including engineered systems, with real-time integration of Oracle's knowledge base with customer environments
- End-to-end performance management and automation
- Integrated Ops Center functionality to monitor and manage both hardware and software from a single interface
- Common management practices applicable to on-premises targets and Oracle Cloud targets
- Quality of Service (QoS) management to ensure delivery of the best service possible to internal and external customers
- Lifecycle and cloud management for simplified provisioning and patching of applications and platforms
- Data governance and compliance controls for conforming with internal and external standards and requirements
- Ability to back up to the cloud to leverage the Oracle Cloud capacity
- Hybrid cloud option to move workloads and clone targets between on-premises and Oracle Cloud

Oracle Database Cloud Service Tools



Oracle Database Cloud Service includes:

- Oracle Database tools such as SQL*Plus and Enterprise Manager Database Express
- Cloud tooling that simplifies backup, recovery, patching, and upgrade operations



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In addition to standard Oracle Database tools such as SQL*Plus, Oracle Database Cloud Service includes cloud tooling that simplifies backup, recovery, patching, and upgrade operations.

Cloud Tooling



Feature	Description	Tools
Simple Automated Backups	Perform on-demand backups Change automatic backup configuration	<code>bkup_api</code> utility <code>raccli</code> utility
Simple Automated Recovery	Restore from backups and recover the database	<code>orec</code> subcommand of the <code>dbaascli</code> utility <code>raccli</code> utility
Simple Automated Patching	Apply patches	<code>dbpatchm</code> subcommand of the <code>dbaascli</code> utility <code>dbpatchmdg</code> utility <code>raccli</code> utility



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Database Cloud Service offers the following tools on the compute nodes associated with the database deployment:

- **Simple Automated Backups:** Use the `bkup_api` utility on a single-instance database or the `raccli` utility on deployments that use Oracle Real Application Clusters (RAC) to perform on-demand backups and change how automatic backups are configured.
- **Simple Automated Recovery:** Use the `orec` subcommand of the `dbaascli` utility on single-instance databases or the `raccli` utility on deployments that use Oracle RAC to restore from backups and recover the database.
- **Simple Automated Patching:** Use the `dbpatchm` subcommand of the `dbaascli` utility for single-instance databases, `raccli` on deployments that use Oracle RAC, and the `dbpatchmdg` utility on Oracle Data Guard configurations to apply patches.

Accessing Tools and Features from the DBCS Console



From the database deployment menu, you can select the following to access consoles and perform other actions:

- Application Express Console
- DBaaS Monitor Console
- EM Console
- SSH Access
- Access Rules
- Delete

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Oracle Database Cloud Service console enables you to manage your database deployment through the following consoles and features:

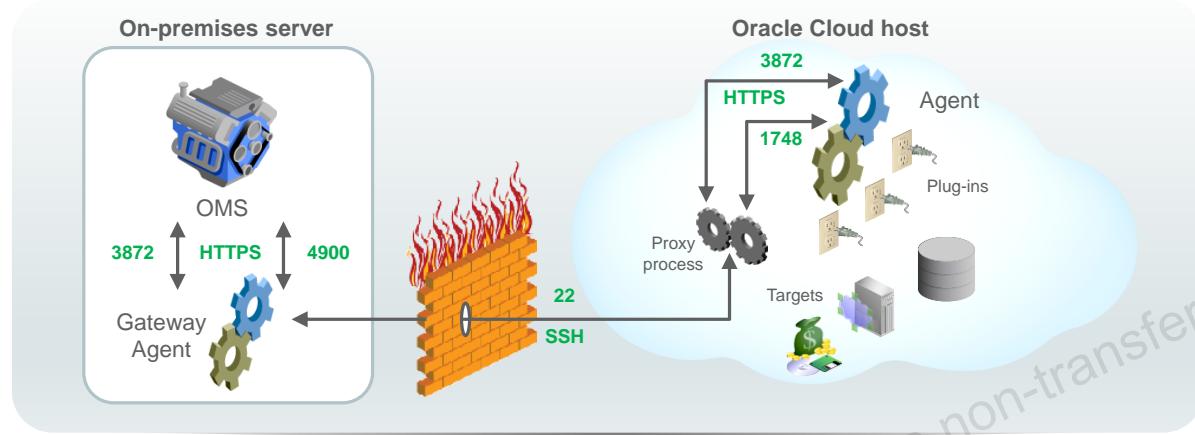
- **Application Express:** Oracle Application Express enables you to design, develop, and deploy responsive database-driven applications by using only your web browser.
- **DBaaS Monitor:** Database deployments include Oracle DBaaS Monitor Console, a built-in monitor that provides a wide spectrum of information about Oracle Database status and resource usage.
- **EM:** To monitor and manage the Oracle database deployed on Database Cloud Service, you can use the standard management tool provided with the version of the database:
 - Enterprise Manager 11g Database Control for Oracle Database 11g
 - Enterprise Manager Database Express 12c for Oracle Database 12c
 - Enterprise Manager Database Express 18c for Oracle Database 18c
- **SSH Access:** Should the need arise, you can add SSH public keys for the `opc` and `oracle` users to the compute nodes associated with a database deployment.
- **Access Rules:** Oracle Database Cloud Service relies on Oracle Compute Cloud Service to provide secure network access to database deployments. You can use the Oracle Database Cloud Service console to perform network access operations such as enabling access to a port on a compute node or creating new security rules.
- **Delete:** When you no longer require a database deployment on Database as a Service, you can delete it. After it is deleted, the entry is removed from the list of database deployments displayed on the Database Cloud Service Console.

See *Administering Oracle Database Cloud Service* for details.

Using Enterprise Manager Cloud Control



You can monitor and manage DBCS with on-premises Enterprise Manager Cloud Control.



See [Using Oracle Enterprise Manager Cloud Control with Database Cloud Service](#) in *Using Oracle Database Cloud Service*

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can use Enterprise Manager Cloud Control 13c to manage and monitor your cloud services and deployments, just as you do the hosts and applications in your own data centers. Cloud Control agents can be deployed to the compute nodes that underpin the Oracle Cloud services. The same plug-ins that manage and monitor your local targets can be used with targets running on the compute nodes. You can install the Enterprise Manager Cloud Control management agent, referred to as the Hybrid Cloud Agent, on the compute node that is deployed for you as part of DBCS.

In order for an Oracle Management Service (OMS) to manage and monitor a host and its targets on the other side of a firewall, directional holes corresponding to the communications between OMS and targets need to be opened in the firewall to allow full functionality. Depending upon the targets being monitored, this can result in a large number of holes in the firewall, and such a topology is anathema to basic security principles and the purpose of firewalls.

The Hybrid Cloud Agent resolves this topological dilemma for hosts in the Oracle Cloud in several ways by using one tunnel for all traffic:

- A Secure Shell (SSH) tunnel is configured in the OMS by providing named SSH credentials for the Oracle Cloud
- When the OMS initiates communication with the agent in Oracle Cloud, it does so via the SSH tunnel by using an EMCTL dispatcher on the Oracle Cloud host
- When the Agent initiates communication with the OMS, it does so via a proxy process that, in turn, communicates via the SSH tunnel with an on-premises agent configured as a gateway agent
- When the OMS initiates direct communication with any of the targets on the Oracle Cloud managed host, such as a database, it does so via the SSH tunnel

Summary

In this lesson, you should have learned how to:

- Connect to an Oracle Database
- Describe the tools used to access an Oracle Database
- Describe the Oracle-supplied user accounts for Oracle Database
- Query the data dictionary in an Oracle Database



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 5: Overview

- 5-1: Viewing Database Deployment Information and Menus
- 5-2: Connecting to the Database Deployment Compute Node
- 5-3: Exploring a CDB by Using SQL*Plus
- 5-4: Exploring a PDB by Using SQL*Plus
- 5-5: Installing the HR Sample Schema
- 5-6: Copying Course Practice Files



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

6

Managing DBCS Database Deployments

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Manage the compute node associated with a database deployment
- Manage network access to a database deployment
- Scale the compute shape and storage
- Patch the database deployment



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Managing the Compute Node

Action	Result of the Action
Start	An Oracle Compute Cloud Service instance is allocated, resources are attached, and it is started.
Stop	An Oracle Compute Cloud Service instance is stopped. Only actions after stop are start and delete.
Restart	An Oracle Compute Cloud Service instance is stopped and immediately started again.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

From the Oracle Database Cloud Service console, you can stop, start, and restart the compute nodes associated with a database deployment. See “Stopping, Starting and Restarting a Database Deployment” in *Administering Oracle Database Cloud Service* for additional information.

Managing Network Access to DBCS (OCI Classic)

- By default, network access to the compute node is provided by Secure Shell (SSH) connections on port 22.
- To access network protocols and services by using a port other than port 22:
 - Use the Oracle Compute Cloud Service console or the Oracle Database Cloud Service console to enable an access rule
 - Create an SSH tunnel to the port



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

By default, network access to the compute nodes associated with Database Cloud Service is provided by Secure Shell (SSH) connections on port 22.

To access network protocols and services on a compute node by using a port other than port 22, you must do one of the following:

- **Enable network access to the port:** You can use the Oracle Compute Cloud Service console or the Oracle Database Cloud Service console to enable access to a port on a compute node by enabling an access rule. See “Enabling Access to a Compute Node Port” in *Administering Oracle Database Cloud Service* for details. **Important Note:** When you enable one of the predefined access rules, the given port on the compute node is opened to the public Internet.
- **Create an SSH tunnel to the port:** Creating an SSH tunnel enables you to access a specific compute node port by using an SSH connection as the transport mechanism. To create the tunnel, you must have the SSH private key file that matches the public key specified during the database deployment creation process. See “Creating an SSH Tunnel to a Compute Node Port” in *Administering Oracle Database Cloud Service* for additional information.

Enabling Access to a Compute Node Port (OCI Classic)

- Database Cloud Service relies on Oracle Compute Cloud Service to provide secure network access to database deployments.
- The Oracle Compute Cloud Service security rules that are created, but not enabled, are as follows:
 - ora_p2_dbconsole: For port 1158, used by Enterprise Manager 11g Database Control
 - ora_p2_dbexpress: For port 5500, used by Enterprise Manager Database Express 12c
 - ora_p2_dblistener: For port 1521, used by SQL*Net
 - ora_p2_http: For port 80, used for HTTP connections
 - ora_p2_httpsl: For port 443, used for HTTPS connections including Oracle REST Data Services, Oracle Application Express, and Oracle DBaaS Monitor
- You enable the security rules through the Database Cloud Service console.
- Enabling one of the predefined security rules opens the given port to the public Internet.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Database Cloud Service relies on Oracle Compute Cloud Service to provide secure network access to database deployments. You can use the Oracle Compute Cloud Service console or the Oracle Database Cloud Service console to perform network access operations, such as enabling access to a port on a compute node.

When a database deployment is created, the Oracle Compute Cloud Service security rules listed in the slide are created, but not enabled.

To enable access to a compute node port, you enable the appropriate security rule. When you enable one of the predefined security rules, the given port on the compute node is opened to the public Internet.

If you want to enable access to a compute node port that is not associated with an existing security rule, you must create a new security rule to define the protocol associated with the port number and create a security rule.

If you want to restrict access to a compute node port, to only permit connections from specific IP addresses, you must create a Security IP List and associate it to the security rule.

See “Enabling Access to a Compute Node Port” in *Administering Oracle Database Cloud Service* for additional information.

Scaling a Database Deployment

Action	Description
Scale Up	Select a new compute shape. Add raw storage to the database deployment.
Scale Down	Select a new compute shape.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If a database deployment on Database Cloud Service is performing poorly or is running out of storage, you can scale up the environment supporting the database deployment.

Scaling the Compute Shape (OCI Classic Only)

When you scale the compute shape of a database deployment on Database Cloud Service, the deployment is put into Maintenance status during the operation, and it is restarted. As a result of the restarting, any resources that you manually added by using the Compute Cloud Service console become detached from the database deployment.

Scaling Storage

When you scale up the storage for a database deployment on Database Cloud Service, the deployment is put into Maintenance status during the operation, and it is restarted. As a result of the restarting, any resources that you manually added to the database deployment by using the Compute Cloud Service console become detached from the deployment.

When you scale up the storage for a database deployment, a Compute Cloud Service storage volume is created and attached to the deployment. This storage volume remains attached and available to the deployment even after it is restarted or is stopped and then started. Also, the storage volume exists until you delete the database deployment, at which time the storage volume is also deleted.

See “Scaling a Database Deployment” in *Administering Oracle Database Cloud Service* for additional information.

Patching DBCS

- Patch management tasks:
 - Check prerequisites before applying a patch.
 - Apply a patch.
 - Roll back a patch or a failed patch.
- Tools and utilities:
 - Oracle Database Cloud Service console for single-instance databases
 - `dbpatchm` subcommand of the `dbaascli` utility for single-instance databases
 - `raccli` utility for Oracle RAC databases
 - `dbpatchmdg` utility for Oracle Data Guard configurations



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can perform several patch management tasks by using the Oracle Database Cloud Service console or command-line interfaces provided with Oracle Database Cloud Service. The tool you use depends on the type of databases that you configured in your database deployment.

Using the DBCS Console to Manage Patches

Use the menu on the Patching tab to:

- Check the prerequisites of a patch before you apply it to a database deployment on Database Cloud Service
- Apply a patch to a database deployment on Database Cloud Service



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can use the Oracle Database Cloud Service Instance Administration page on database deployments hosting single-instance databases to do the following:

- Check the prerequisites of a patch before you apply it to a database deployment. See “Checking Prerequisites Before Applying a Patch” in *Administering Oracle Database Cloud Service* for details.
- Apply a patch to a database deployment. See “Applying a Patch” in *Administering Oracle Database Cloud Service*.

Using the `dbaascli` Utility to Manage Patches

- Checking prerequisites:

```
# dbaascli dbpatchm --run -prereq
```

- Applying a patch:

```
# dbaascli dbpatchm --run -apply
```

- Rolling back a patch:

```
# dbaascli dbpatchm --run -rollback
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can use the `dbpatchm` subcommand of the `dbaascli` utility on database deployments hosting single-instance databases to:

- Check the prerequisites of a patch before you apply it to a database deployment. See “Checking Prerequisites Before Applying a Patch” in *Administering Oracle Database Cloud Service* for additional information.
- Apply a patch to a database deployment. See “Applying a Patch” in *Administering Oracle Database Cloud Service* for details.
- Roll back the last patch or failed patch attempt to a database deployment. Before executing the `dbpatchm` subcommand to roll back a patch, you must edit the `/var/opt/oracle/patch/rollbackpatches.txt` file to add patch information. See “Rolling Back a Patch or Failed Patch” in *Administering Oracle Database Cloud Service* for additional information.

The DBCS console does not currently reflect patching information if you apply a patch by using a command-line utility on a database deployment’s compute nodes.

Summary

In this lesson, you should have learned how to:

- Manage the compute node associated with a database deployment
- Manage network access to a database deployment
- Scale the compute shape and storage
- Patch the database deployment



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 6: Overview

- 6-1: Accessing Enterprise Manager Database Express
- 6-2: Exploring a CDB and PDB by Using Enterprise Manager Database Express



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Managing Database Instances

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

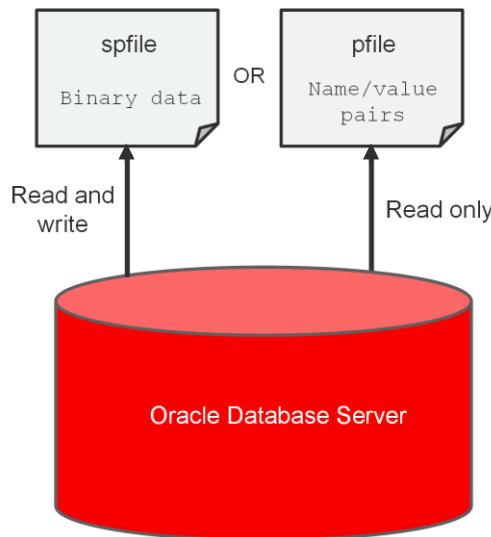
- Describe initialization parameter files and initialization parameters
- View and modify initialization parameters in SQL*Plus
- Start up and shut down Oracle databases
- Open and close PDBs
- Work with the Automatic Diagnostic Repository (ADR)
- Query dynamic performance views



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Working with Initialization Parameters



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Initialization Parameter Files

When you start a database instance, it reads instance configuration parameters (initialization parameters) from an initialization parameter file (parameter file). On most platforms, parameter files are stored in the \$ORACLE_HOME/dbs directory by default.

You can use one of the following types of parameter files to start your database instance, as illustrated in the slide:

- **Server parameter file (SPFILE)**: An SPFILE is a binary file that is written to and read by the database server. You can't edit it manually. An SPFILE is preferred over a PFILE because you can change initialization parameters with `ALTER SYSTEM` commands in SQL*Plus, and the changes persist when you shut down and start up the database instance. It also provides a basis for self-tuning by Oracle Database. An SPFILE is automatically created for you by Database Configuration Assistant (DBCA) when you create a CDB. It resides on the server on which the Oracle instance is running. The default name of the SPFILE, which is automatically sought at startup, is `SPFILE<SID>.ora`.
- **Text initialization parameter file (PFILE)**: A PFILE is a text file containing parameter values in name/value pairs, which the database server can read to start the database instance. Unlike an SPFILE, the database server cannot write to and alter a PFILE. Therefore, to change parameter values in a PFILE and make them persist during shutdown and startup, you must manually edit the PFILE in a text editor and restart the database instance to refresh the parameter values. Your installation includes a sample PFILE named `init.ora` in the default directory for parameter files. You can use this file as a starting point for a PFILE, or you can create a PFILE from the SPFILE. If you save your PFILE as `init<SID>.ora` in the default directory, the database server will automatically use it if an SPFILE is not available. If you save the PFILE under a different name, you'll need to specify it during startup.

Search Order for a Parameter File

The database server locates your parameter file by examining file names in the \$ORACLE_HOME/dbs directory in the following order:

1. SPFILE<SID>.ora, where SID is the system ID and identifies the instance name (for example, ORCL)
2. SPFILE.ora
3. init<SID>.ora (PFILE)

Initialization Parameters

- Initialization parameters (parameters):
 - Set database limits
 - Set database-wide defaults
 - Specify files and directories
 - Affect performance
- Parameters can be of two types: basic or advanced.
 - Tune around 30 basic parameters to get reasonable database performance.
 - Example of a basic parameter: `SGA_TARGET`
 - Example of an advanced parameter: `DB_CACHE_SIZE`
- Derived parameters calculate their values from the values of other parameters.
 - Example: `SESSIONS` is derived from `PROCESSES`.
- Some parameter values or value ranges depend on the host operating system.
 - Example: `DB_BLOCK_SIZE`



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Initialization parameters (parameters) set database limits, set databasewide defaults, specify files and directories, and affect performance. The parameter file must, at a minimum, specify the `DB_NAME` parameter. All other parameters have default values.

Types of Initialization Parameters

Parameters can be of two types: basic or advanced. In the majority of cases, you'll need to set and tune only the 30 or so basic parameters to get reasonable performance from the database. In rare situations, you'll need to modify one or more of the 300 or so advanced parameters to achieve optimal performance. An example of a basic parameter is `SGA_TARGET`, which specifies the total memory size of all SGA components. And example of an advanced parameter is `DB_CACHE_SIZE`, which specifies the size of the default buffer pool.

Derived Parameters

Some parameters are derived, meaning their values are calculated from the values of other parameters. Normally, you shouldn't alter values for derived parameters. But if you do, the value that you specify overrides the calculated value. For example, the default value of the `SESSIONS` parameter is derived from the value of the `PROCESSES` parameter. If the value of `PROCESSES` changes, the default value of `SESSIONS` changes as well, unless you override it with a specified value.

Parameter Values That Depend on the OS

Some parameter values or value ranges depend on the host operating system. For example, the `DB_FILE_MULTIBLOCK_READ_COUNT` parameter specifies the maximum number of blocks that are read in one I/O operation during a sequential scan; this parameter is platform dependent. The size of those blocks, which is set by `DB_BLOCK_SIZE`, has a default value that depends on the operating system.

View

Review the information about the `CONTROL_FILES` parameter. *Oracle Database Reference* is a good source of information about parameters. In this example, you'll learn the parameter's data type (string), syntax (`CONTROL_FILES = file name, [, file name]...`), default value (operating system-dependent), whether its modifiable (no), whether you can modify its value in a PDB (no), its range of values (1 to 8 file names), whether it is a basic parameter (yes), and details for Oracle Real Application Clusters (multiple instances must have the same value).

Modifying Initialization Parameters

- Modify parameters to set capacity limits or improve performance.
 - Use EM Express or SQL*Plus (ALTER SESSION or ALTER SYSTEM).
- Query V\$PARAMETER for an initialization parameter to learn whether you can make:
 - Session-level changes (ISSES_MODIFIABLE column)
 - System-level changes (ISSYS_MODIFIABLE column)
 - PDB-level changes (ISPDB_MODIFIABLE column)
- Use the SCOPE clause with the ALTER SYSTEM command to tell the system where to update the system-level parameter:
 - MEMORY
 - SPFILE
 - BOTH
- Use the DEFERRED keyword to set or modify the value of the parameter for future sessions that connect to the database.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Using ALTER SESSION or ALTER SYSTEM Commands

You modify parameters because you want to set capacity limits or improve performance. You can use the ALTER SESSION or ALTER SYSTEM commands in SQL*Plus to modify parameters. In your own environment, you'll likely only modify the basic initialization parameters to keep your database running with good performance. You can also use EM Express to modify parameters.

Increasing the values of parameters may improve your system's performance, but increasing most parameters also increases the SGA size. A larger SGA can improve database performance up to a point. An SGA that is too large can degrade performance if it is swapped in and out of memory. You should set operating system parameters that control virtual memory working areas with the SGA size in mind. The operating system configuration can also limit the maximum size of the SGA.

Before modifying a parameter, you should query the V\$PARAMETER view to learn about how you can modify a parameter.

- The ISSES_MODIFIABLE column value tells you whether you can change the parameter for your current session (TRUE) or not (FALSE) by using the ALTER SESSION command. You can change some parameters at the session level, but not all. Changes are applied to your current session immediately (dynamically) and expire when you end your session. Parameters with a value of TRUE are referred to as session-level parameters.

Example: SQL> ALTER SESSION SET NLS_DATE_FORMAT ='mon dd yyyy';

- The ISSYS_MODIFIABLE column value tells you when a system-level change to the parameter, made by using the ALTER SYSTEM command, takes effect.
 - IMMEDIATE means the change will take effect immediately and be applied to all current sessions.
 - DEFERRED means the change will take effect in subsequent sessions.
 - FALSE means the change will take effect in subsequent instances.

You can change all parameters at the system level by using the ALTER SYSTEM command, and the change is applied to all sessions.

Parameters with a value of FALSE are referred to as static parameters. For static parameters, you need to shut down and restart the database instance to implement the change. Also, the database instance must have been started with an SPFILE.

Example: SQL> ALTER SYSTEM SET SEC_MAX_FAILED_LOGIN_ATTEMPTS=2 SCOPE=SPFILE;

- The ISPDB_MODIFIABLE column value tells you whether you can (TRUE) or can't (FALSE) modify the parameter inside a PDB. In a non-CDB, the value of this column is NULL.

Setting the Scope in the ALTER SYSTEM Command

Use the SCOPE clause with the ALTER SYSTEM command to tell the system where to update the system-level parameter. This location dictates how long the change will stay in effect. Scope also depends on whether you started the database instance by using a PFILE or an SPFILE. Scope can have the following values:

- MEMORY: This value tells the system to make the parameter change in memory only. The change will take effect immediately, but not persist in subsequent sessions. If you started the database instance by using a PFILE, then this is the only scope you can specify. This specification is not allowed for static parameters.
- SPFILE: This value tells the system to make the parameter change in the SPFILE only. The change will take effect immediately and persist after you restart the database instance. This is the only scope allowed for static parameters.
- BOTH: This value tells the system to make the parameter change in both memory and in the SPFILE. The change will take effect immediately and persist after you restart the database instance. If you started the database instance by using an SPFILE, then BOTH is the default.

Using the DEFERRED Keyword

The DEFERRED keyword tells the system to make the parameter change effective only for future sessions. You must specify DEFERRED in the ALTER SYSTEM command if the value of the ISSYS_MODIFIABLE column is DEFERRED. If the value of that column is IMMEDIATE, then the DEFERRED keyword is optional. If the value of that column is FALSE, then you cannot specify DEFERRED in the ALTER SYSTEM statement.

Viewing Initialization Parameters

Ways to view initialization parameters in SQL*Plus:

- Issue the SHOW PARAMETER command.
 - Example: Find out about all the parameters whose names contain the word “para.”

```
SQL> SHOW PARAMETER para
```
- Query the following views:
 - V\$PARAMETER
 - V\$PARAMETER2
 - V\$SPPARAMETER
 - V\$SYSTEM_PARAMETER
 - V\$SYSTEM_PARAMETER2



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Issuing the SHOW PARAMETER Command

You can issue the SHOW PARAMETER command in SQL*Plus to view information about an initialization parameter (for example, view a parameter's data type and default value). For instance, the following SHOW PARAMETER command returns information about parameters whose names contain the word “para.”

```
SQL> SHOW PARAMETER para
```

NAME	TYPE	VALUE
<hr/>		
cell_offload_parameters	string	
fast_start_parallel_rollback	string	LOW
parallel_adaptive_multi_user	boolean	TRUE
parallel_automatic_tuning	boolean	FALSE

Querying Views

You can also query the V\$PARAMETER view in SQL*Plus to view information about an initialization parameter. For example, the following query against the V\$PARAMETER view returns information about parameters whose names contain the word “pool.”

SQL> SELECT name, value FROM v\$parameter WHERE name LIKE '%pool%';

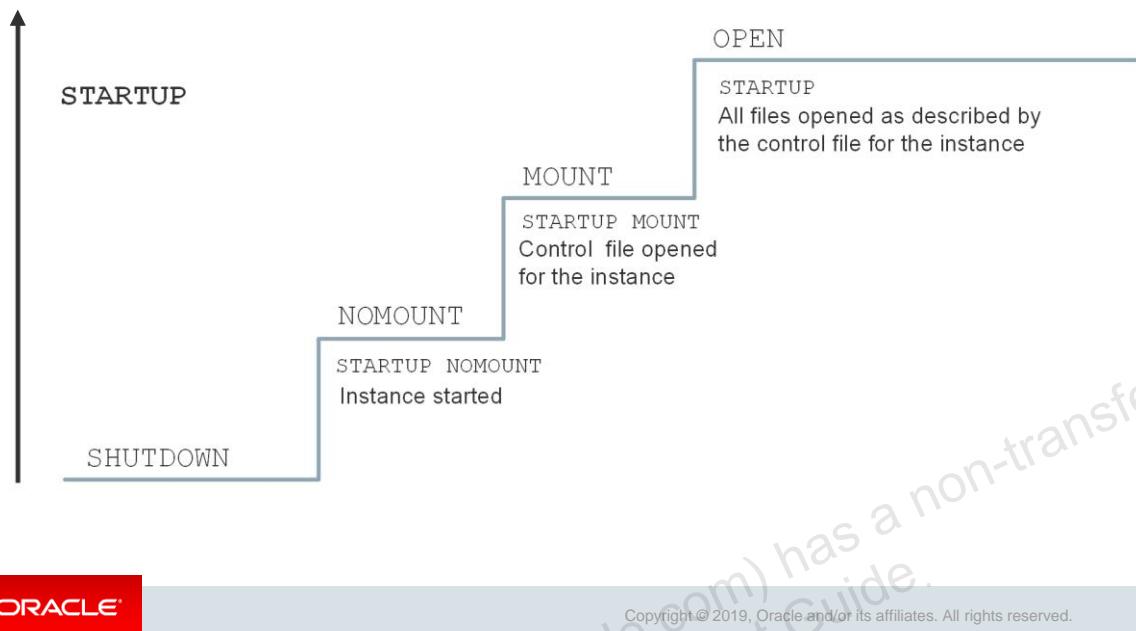
NAME	VALUE
shared_pool_size	0
large_pool_size	0
java_pool_size	0
streams_pool_size	0
shared_pool_reserved_size	15728640
...	

9 rows selected.

Other views that contain parameter information include:

- V\$SPPARAMETER: Displays information about the contents of the SPFILE. If you didn't use an SPFILE to start the database instance, each row of the view will contain FALSE in the ISSPECIFIED column.
- V\$PARAMETER2: Displays information about the parameters that are currently in effect for the session, with each parameter value appearing as a row in the view. A new session inherits parameter values from the database instance-wide values displayed in the V\$SYSTEM_PARAMETER2 view.
- V\$SYSTEM_PARAMETER: Displays information about the parameters that are currently in effect for the database instance.
- V\$SYSTEM_PARAMETER2: Displays information about the initialization parameters that are currently in effect for the instance, with each list parameter value appearing as a row in the view.

Starting the Oracle Database Instance



Before users can connect to a database instance, a database administrator must start the database instance. The database instance and database go through stages as the database is made available for access by users. The database instance is started, the database is mounted, and then the database is opened, as shown in the slide.

You can use the `STARTUP` command in SQL*Plus with the options shown in the slide for each stage. The default option is `OPEN`.

NOMOUNT: During this stage, the Oracle software reads an initialization parameter file, starts background processes, allocates memory to the SGA, and opens the alert log and trace files. An instance is typically started only in `NOMOUNT` mode during database creation, during re-creation of control files, or in certain backup and recovery scenarios.

MOUNT: During this stage, the Oracle software associates the database (CDB) with the previously started database instance, opens and reads the control files that are specified in the initialization parameter file, and obtains the names and statuses of the data files and online redo log files. No checks, however, are performed to verify the existence of the data files and online redo log files at this time. Start in `MOUNT` mode to perform some maintenance operations, such as renaming data files and performing full database recoveries.

OPEN: The Oracle software opens the redo log files and data files according to the list registered in the control files. Start up in `OPEN` mode to enable users to connect to the database instance. PDBs are not, by default, started when you open the database.

Shutting Down an Oracle Database Instance

- Sometimes you need to shut down the database instance (for example, to change a static parameter or patch the database server).
- Use the `SHUTDOWN` command to shut down the database instance in various modes: `ABORT`, `IMMEDIATE`, `TRANSACTIONAL`, and `NORMAL`.

	ABORT	IMMEDIATE	TRANSACTIONAL	NORMAL
Allows new connections	No	No	No	No
Waits until current sessions end	No	No	No	Yes
Waits until current transactions end	No	No	Yes	Yes
Forces a checkpoint and closes files	No	Yes	Yes	Yes



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

IMMEDIATE Mode: A shutdown in `IMMEDIATE` mode is the most typically used option.

- Current SQL statements being processed by the database instance are not completed.
- The database server does not wait for the users who are currently connected to the database instance to disconnect.
- The database server rolls back active transactions and disconnects all connected users.
- The database server closes and dismounts the database before shutting down the database instance.

TRANSACTIONAL Mode: A shutdown in `TRANSACTIONAL` mode prevents clients from losing data, including results from their current activity.

- No client can start a new transaction on this particular instance.
- A client is disconnected when the client ends the transaction that is in progress.
- When all transactions have been completed, a shutdown occurs immediately.

NORMAL Mode: `NORMAL` is the default shutdown mode if no mode is specified with the `SHUTDOWN` command.

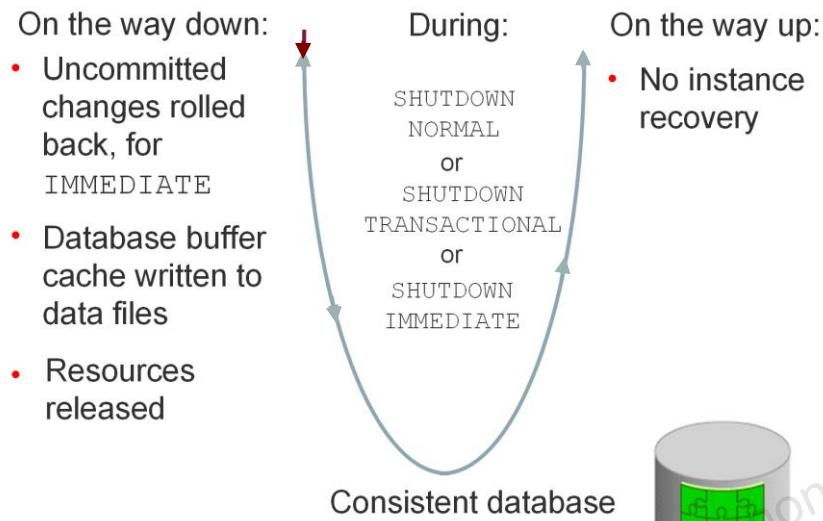
- No new connections can be made.
- The Oracle server waits for all users to disconnect before completing the shutdown.
- Database and redo buffers are written to disk.
- Background processes are terminated, and the SGA is removed from memory.
- The Oracle server closes and dismounts the database before shutting down the instance.

ABORT Mode: If the other shutdown modes don't work, you can use ABORT mode. ABORT mode performs the least amount of work before shutting down. Because this mode puts the database in an inconsistent state and requires recovery before startup, use it only when necessary. It's not advisable to back up the database in this state. It's typically used when no other form of shutdown works, when there are problems with starting the database instance, or when you need to shut down immediately because of an impending situation (such as notice of a power outage within seconds). ABORT mode is usually the fastest shutdown mode and NORMAL mode is the slowest. NORMAL and TRANSACTIONAL modes can take a long time depending on the number of sessions and transactions.

The following happens during a shutdown in ABORT mode, an instance failure, or a database instance startup in FORCE mode:

- Current SQL statements being processed by the Oracle server are immediately terminated.
- The Oracle server does not wait for users who are currently connected to the database to disconnect.
- Database and redo buffers are not written to disk.
- Uncommitted transactions are not rolled back.
- The instance is terminated without closing the files.
- The database is not closed or dismounted.
- The next startup requires instance recovery, which occurs automatically.

Comparing SHUTDOWN Modes

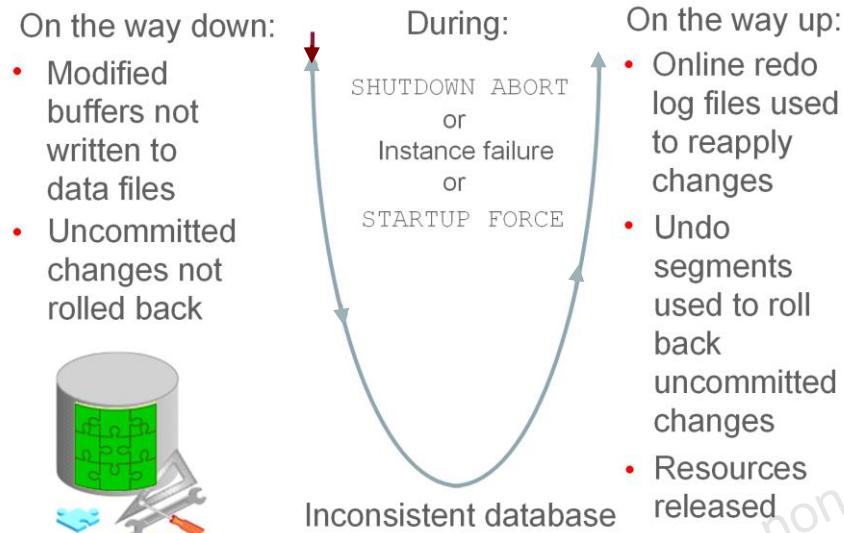


ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The diagram in this slide shows the **IMMEDIATE**, **TRANSACTIONAL**, and **NORMAL** shutdown modes. The diagram in the next slide shows **ABORT** shutdown mode (and instance failure or **STARTUP FORCE** mode). Notice that the database becomes inconsistent when you perform an **ABORT** shutdown, whereas it stays consistent during the other shutdown modes. Also note that you need to recover the database instance after you perform an **ABORT** shutdown; whereas with the other shutdown modes, you don't need to do so.

Comparing SHUTDOWN Modes



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Notice that the database becomes inconsistent when you perform an **ABORT** shutdown, whereas it stays consistent during the other shutdown modes. Also note that you need to recover the database instance after you perform an **ABORT** shutdown; whereas with the other shutdown modes, you don't need to do so.

Opening and Closing PDBs

- Open/close a PDB to open/close its data files.
- A PDB has four open modes:
 - READ WRITE (the PDB is fully started/opened)
 - READ ONLY
 - MIGRATE
 - MOUNTED (the PDB is shut down/closed)
- Use the ALTER PLUGGABLE DATABASE command or STARTUP and SHUTDOWN commands to open and close PDBs.
 - Example: SQL> ALTER PLUGGABLE DATABASE PDB1 OPEN;
- The ALTER PLUGGABLE DATABASE command lets you change from any open mode to another.
- To use the STARTUP command, the PDB must be in MOUNTED mode.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Open Modes

Starting up a PDB and opening a PDB mean the same thing, and you'll find both phrases used in documentation and online resources. When you open a PDB, the database server opens the data files for that PDB. Similar to a CDB, a PDB has four levels of being open, and these levels are referred to as open modes. The open modes are READ WRITE (the PDB is fully started/opened), READ ONLY, MIGRATE, and MOUNTED (the PDB is shut down/closed).

Commands to Open and Close PDBs

You can use the ALTER PLUGGABLE DATABASE command to open and close a PDB from either the root container or within the PDB itself. You can also use STARTUP and SHUTDOWN commands. The ALTER PLUGGABLE DATABASE command lets you change from any open mode to another for a PDB. To use the STARTUP command, the PDB must first be in MOUNTED mode. Either command requires you to be connected to the root container or PDB with one of the following system privileges: AS SYSBACKUP, AS SYSDBA, AS SYSDG, or AS SYSOPER.

Examples

In this example, PDB1 is started up (opened). Its open mode is changed from MOUNT to READ WRITE.

```
SQL> ALTER PLUGGABLE DATABASE PDB1 OPEN;
```

In this example, PDB1 is shut down (closed). Its open mode is changed to MOUNT.

```
SQL> ALTER PLUGGABLE DATABASE PDB1 CLOSE;
```

Working with the Automatic Diagnostic Repository

The Automatic Diagnostic Repository (ADR):

- Is a file-based repository outside the database
- Is a system-wide central tracing and logging repository
- Stores database diagnostic data such as:
 - Traces
 - Alert log
 - Health monitor reports



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Automatic Diagnostic Repository

The Automatic Diagnostic Repository (ADR) is a system-wide tracing and logging central repository for database diagnostic data such as traces, the alert log, health monitor reports, and more.

The ADR root directory is known as the ADR base. Its location is set by the `DIAGNOSTIC_DEST` initialization parameter (for example, `/u01/app/oracle`).

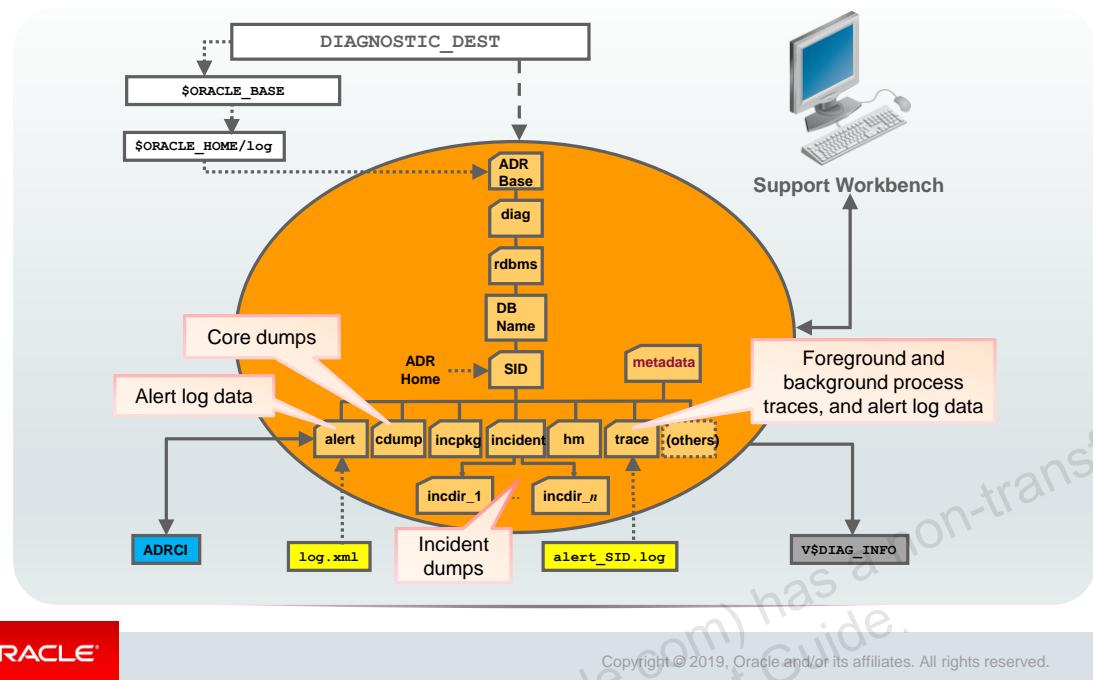
The location of an ADR home is given by the following path, which starts at the ADR base directory:

`<ADR Base>/diag/product_type/db_id/instance_id`

For example:

`/u01/app/oracle/diag/rdbms/orcl/ORCL`

Automatic Diagnostic Repository



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

ADR is a file-based repository for database diagnostic data such as traces, incident dumps and packages, the alert log, Health Monitor reports, core dumps, and more. It has a unified directory structure across multiple instances and multiple products—stored outside of any database. It is, therefore, available for problem diagnosis when the database is down.

The Oracle Database server, Automatic Storage Management (ASM), Cluster Ready Services (CRS), and other Oracle products or components store all diagnostic data in the ADR. Each instance of each product stores diagnostic data underneath its own ADR home directory. For example, in a Real Application Clusters environment with shared storage and ASM, each database instance and each ASM instance have a home directory within the ADR. ADR's unified directory structure, consistent diagnostic data formats across products and instances, and a unified set of tools enable customers and Oracle Support to correlate and analyze diagnostic data across multiple instances.

The ADR root directory is known as the ADR base. Its location is set by the `DIAGNOSTIC_DEST` initialization parameter. If this parameter is omitted or left null, the database sets `DIAGNOSTIC_DEST` upon startup as follows: If environment variable `ORACLE_BASE` is set, `DIAGNOSTIC_DEST` is set to `$ORACLE_BASE`. If environment variable `ORACLE_BASE` is not set, `DIAGNOSTIC_DEST` is set to `$ORACLE_HOME/log`.

Viewing the Alert Log

- The alert log file is a chronological log of messages about the database instance and database, such as:
 - Any nondefault initialization parameters used at startup
 - All internal errors (ORA-600), block corruption errors (ORA-1578), and deadlock errors (ORA-60) that occurred
 - Administrative operations, such as the SQL statements CREATE, ALTER, DROP DATABASE, and TABLESPACE, and the Enterprise Manager or SQL*Plus statements STARTUP, SHUTDOWN, ARCHIVE LOG, and RECOVER
 - Several messages and errors relating to the functions of shared server and dispatcher processes
 - Errors during the automatic refresh of a materialized view
- Query V\$DIAG_INFO to find the location of the alert log.
 - The path to alert_SID.log corresponds to the Diag Trace entry.
 - The path to log.xml corresponds to the Diag Alert entry.
- You can view the alert log in a text editor or in ADRCI.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Each database instance has an alert_SID.log file. The file is on the server with the database and is stored in \$ORACLE_BASE/diag/rdbms/<db_name>/<SID>/trace by default if \$ORACLE_BASE is set.

Oracle Database uses the alert log to keep a record of these events as an alternative to displaying the information on an operator's console. Many systems also display this information on the console. If an administrative operation is successful, a message is written in the alert log as "completed" along with a time stamp.

Enterprise Manager Cloud Control monitors the alert log file and notifies you of critical errors. You can also view the log to see noncritical error and information messages. Because the file can grow to an unmanageable size, you can periodically back up the alert file and delete the current alert file. When the database attempts to write to the alert file again, it creates a new one.

Note: There is an XML version of the alert log in the \$ORACLE_BASE/diag/rdbms/<db_name>/<SID>/alert directory.

ADRCI is an Oracle command-line utility that enables you to investigate problems, view health check reports, and package and upload first-failure data to Oracle Support. You can also use the utility to view the names of the trace files in the Automatic Diagnostic Repository (ADR) and the alert log. ADRCI has a rich command set that you can use interactively or in scripts.

Using Trace Files

- Trace files contain:
 - Error information (contact Oracle Support Services if an internal error occurs)
 - Information that can provide guidance for tuning applications or an instance
- Each server and background process can write to an associated trace file.
- Trace file names for background processes are named after their processes.
 - Exception: Trace files generated by job queue processes
- Oracle Database includes an advanced fault diagnosability infrastructure for preventing, detecting, diagnosing, and resolving problems.
- When a critical error occurs:
 - An incident number is assigned to the error
 - Diagnostic data for the error (such as trace files) is immediately captured and tagged with the incident number
 - Data is stored in the ADR
- ADR files can be automatically purged by setting retention policy parameters.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Trace Files

Each server and background process can write to an associated trace file. When a process detects an internal error, it dumps information about the error to its trace file. If an internal error occurs and information is written to a trace file, the administrator should contact Oracle Support Services.

All file names of trace files associated with a background process contain the name of the process that generated the trace file. The one exception to this is trace files that are generated by job queue processes (Jnnn).

Additional information in trace files can provide guidance for tuning applications or an instance. Background processes always write this information to a trace file when appropriate.

Oracle Database includes an advanced fault diagnosability infrastructure for preventing, detecting, diagnosing, and resolving problems. In particular, problems that are targeted include critical errors such as those caused by database code bugs, metadata corruption, and customer data corruption.

When a critical error occurs, an incident number is assigned to it; diagnostic data for the error (such as trace files) is immediately captured and tagged with this number. The data is then stored in the Automatic Diagnostic Repository (ADR)—a file-based repository outside the database—where it can later be retrieved by incident number and analyzed.

Purging Mechanism

The purging mechanism allows you to specify a retention policy stating:

- How old ADR contents should be before they are automatically deleted.
 - The long retention period is used for the relatively higher-value diagnostic data, such as incidents and alert log (default value is 365 days).
 - The short retention period is used for traces and core dumps (default value is 30 days).
- Older items are deleted first. The long retention period items are typically older than any of the items in the short retention period. So a mechanism is used in which the time periods are “scaled” so that roughly the same percentage of each gets deleted. Some components use these periods in slightly different ways. For instance, IPS, the packaging facility, uses the short retention period to determine when to purge packaging metadata and the staging directory contents. However, the age of the data is based on when the package was completed, not when it was originally created.
- The size-based retention to specify a target size for an ADR home. When purging, the old data, determined by the time-based retention periods, is deleted first. If the size of the ADR home is still greater than the target size, diagnostics are automatically deleted until the target size is no longer exceeded.

Administering the DDL Log File

- Enable the capture of certain DDL statements to a DDL log file by setting `ENABLE_DDL_LOGGING` to TRUE.
- The DDL log contains one log record for each DDL statement.
- Two DDL logs containing the same information:
 - XML DDL log: `log.xml` written to
 `$ORACLE_BASE/diag/rdbms/<dbname>/<SID>/log/ddl`
 - Text DDL: `ddl_<sid>.log` written to
 `$ORACLE_BASE/diag/rdbms/<dbname>/<SID>/log`
- Example:

```
$ more ddl_orcl.log
Thu Nov 15 08:35:47 2016
diag_adl:drop user app_user
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The DDL log is created only if the `ENABLE_DDL_LOGGING` initialization parameter is set to TRUE. When this parameter is set to FALSE, DDL statements are not included in any log. A subset of executed DDL statements is written to the DDL log.

There are two DDL logs that contain the same information. One is an XML file, and the other is a text file. The DDL log is stored in the `log/ddl` subdirectory of the ADR home.

Important: You must have a license for Oracle Database Lifecycle Management Pack to enable DDL logging.

Querying Dynamic Performance Views

- Dynamic performance views provide access to information about the changing states of instance memory structures:
 - Sessions, file states, and locks
 - Progress of jobs and tasks
 - Backup status, memory usage, and allocation
 - System and session parameters
 - SQL execution
 - Statistics and metrics
- Dynamic performance views start with the prefix V\$.
- Example query: Which current sessions have logged in from the EDXX9P1 computer on the last day?

```
SQL> SELECT * FROM V$SESSION
  2 WHERE machine = 'EDXX9P1'
  3 AND logon_time > SYSDATE - 1;
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Oracle Database server maintains a dynamic set of data about the operation and performance of the database instance. The dynamic performance views are based on virtual tables that are built from memory structures inside the database server. They are not conventional tables that reside in a database. This is the reason that some of them are available before a database is mounted or open.

Note: The DICT and DICT_COLUMNS views also contain the names of these dynamic performance views.

You can use dynamic performance views to answer questions such as the following:

1. For which SQL statements (and their associated numbers of executions) is the CPU time consumed greater than 200,000 microseconds?

```
SQL> SELECT sql_text, executions FROM V$SQL WHERE cpu_time >
200000;
```

2. What are the session IDs of those sessions that are currently holding a lock that is blocking another user, and how long have those locks been held?

```
SQL> SELECT sid, ctime FROM v$lock WHERE block > 0;
```

Considerations for Dynamic Performance Views

- These views are owned by the `SYS` user.
- Views provide information depending on the stage (`NOMOUNT`, `MOUNT`, or `OPEN`).
- You can query `V$FIXED_TABLE` to see all the view names.
- These views are often referred to as “v-dollar views.”
- Read consistency is not guaranteed on these views because the data is dynamic.

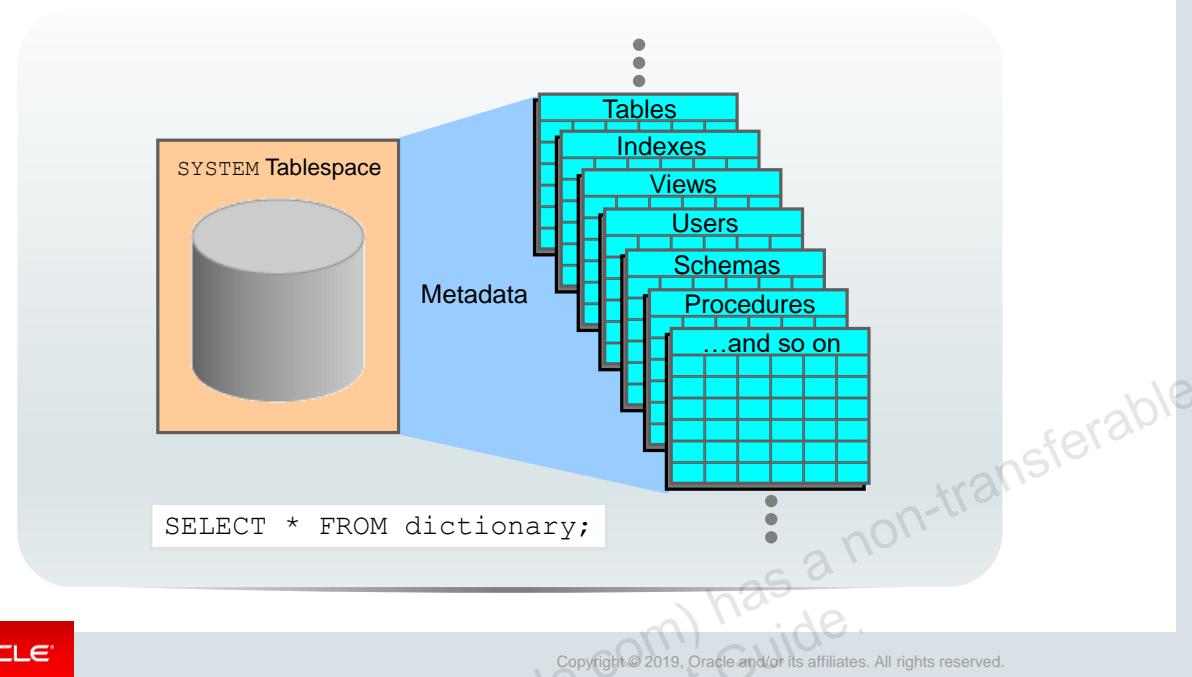


Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Some dynamic views provide information that is not applicable to all states of an instance or database. For example, if an instance has just been started but no database is mounted, you can query `V$BGPROCESS` to see the list of background processes that are running. But querying `V$DATAFILE` to see the status of database data files would return no rows. The database must be mounted or opened for `V$DATAFILE` to provide meaningful information. It is when the database is mounted that the control file is read to obtain information about the data files associated with a database.

Some `V$` views contain information that is similar to information in the corresponding `DBA_` views. For example, `V$DATAFILE` is similar to `DBA_DATA_FILES`. Note also that `V$` view names are generally singular and `DBA_` view names are plural.

Data Dictionary: Overview



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

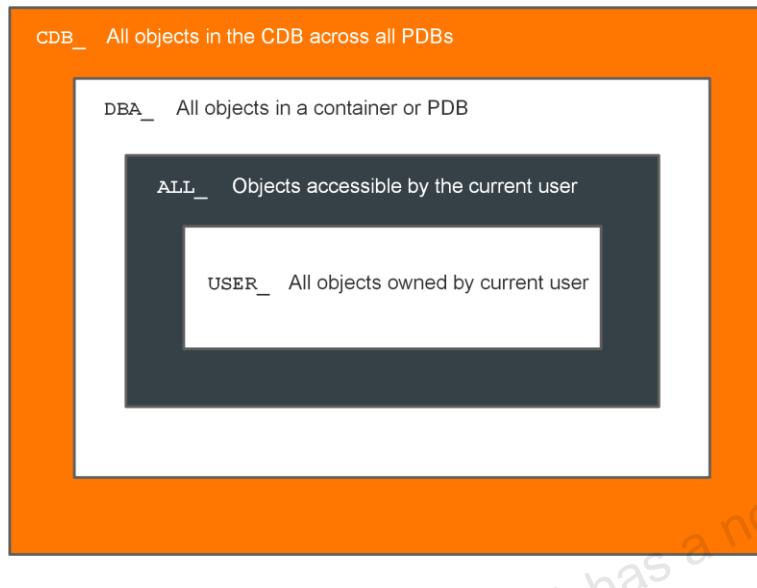
The Oracle data dictionary is the metadata of the database and contains the names and attributes of all objects in the database. The creation or modification of any object causes an update to the data dictionary that reflects those changes. This information is stored in the base tables that are maintained by the Oracle Database server, but you access these tables by using predefined views rather than by reading the tables directly.

The data dictionary:

- Is used by the Oracle Database server to find information about users, objects, constraints, and storage
- Is maintained by the Oracle Database server when object structures or definitions are modified
- Is available for use by any user to query information about the database
- Is owned by the `SYS` user
- Should never be modified directly by using SQL

Note: The `DICTIONARY` data dictionary view (or the `DICT` synonym for this) contains the names and descriptions of data dictionary tables and views. Use the `DICT_COLUMNS` view to see the view columns and their definitions. For complete definitions of each view, see the *Oracle Database Reference*. There are over 1000 views that reference hundreds of base tables.

Querying the Oracle Data Dictionary



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

CDB_, DBA_, ALL_, and USER_ Views

The view prefixes, as shown in the slide, indicate the data (and how much of that data) a given user can see.

- `CDB_` views display metadata for all objects in a CDB across all PDBs.
- `DBA_` views display metadata for all objects in a container or PDB.
- `ALL_` views display metadata for objects that the current user is privileged to see, whether the user owns them or not. For example, if `USER_A` has been granted access to a table owned by `USER_B`, then `USER_A` sees that table listed in any `ALL_` view dealing with table names.
- `USER_` views display metadata for all objects owned by the current user; that is, objects that are present in the user's own schema.

Only `USER_` and `ALL_` views are available to any user. The `CDB_` and `DBA_` views are restricted to DBA accounts.

Generally, each view set is a subset of the higher-privileged view set, row-wise and column-wise. Not all views in a given view set have a corresponding view in the other view sets. It depends on the nature of the information in the view. For example, there is a `DBA_LOCK` view, but no `ALL_LOCK` view, because only a DBA would have interest in data about locks. Be sure to choose the appropriate view set to meet the need that you have. If you have the privilege to access the DBA views, you still may want to query only the `USER_` version of the view because the results show information on objects that you own and you may not want other objects to be added to your result set.

The `CDB_` and `DBA_` views can be queried only by users with the `SYSDBA` or `SELECT ANY DICTIONARY` privilege, or `SELECT_CATALOG_ROLE` role, or by users with direct privileges granted to them.

When a user connected to the root queries a `CDB_*` view, the query results will depend on the `CONTAINER_DATA` attribute for the user. The `CONTAINER_DATA` clause of the SQL `ALTER USER` statement is used to set and modify the users' `CONTAINER_DATA` attribute. In a PDB, the `CDB_*` views only show objects visible through a corresponding `DBA_*` view.

Summary

In this lesson, you should have learned how to:

- Describe initialization parameter files and initialization parameters
- View and modify initialization parameters in SQL*Plus
- Start up and shut down Oracle databases
- Open and close PDBs
- Work with the Automatic Diagnostic Repository (ADR)
- Query dynamic performance views



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 7: Overview

- 7-1: Investigating Initialization Parameter Files
- 7-2: Viewing Initialization Parameters by Using SQL*Plus
- 7-3: Modifying Initialization Parameters by Using SQL*Plus
- 7-4: Modifying an Initialization Parameter by Using Enterprise Manager Database Express
- 7-5: Shutting Down and Starting Up the Oracle Database
- 7-6: Viewing Diagnostic Information



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Oracle Net Services



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Objectives

After completing this lesson, you should be able to:

- Describe Oracle Net Services
- Explain how listeners work
- Configure listeners for dynamic or static service registration
- Configure local naming for database connections
- Test Oracle Net connectivity with `tnsping`
- Configure communication between databases by creating database links
- Explain the difference between dedicated and shared server configurations

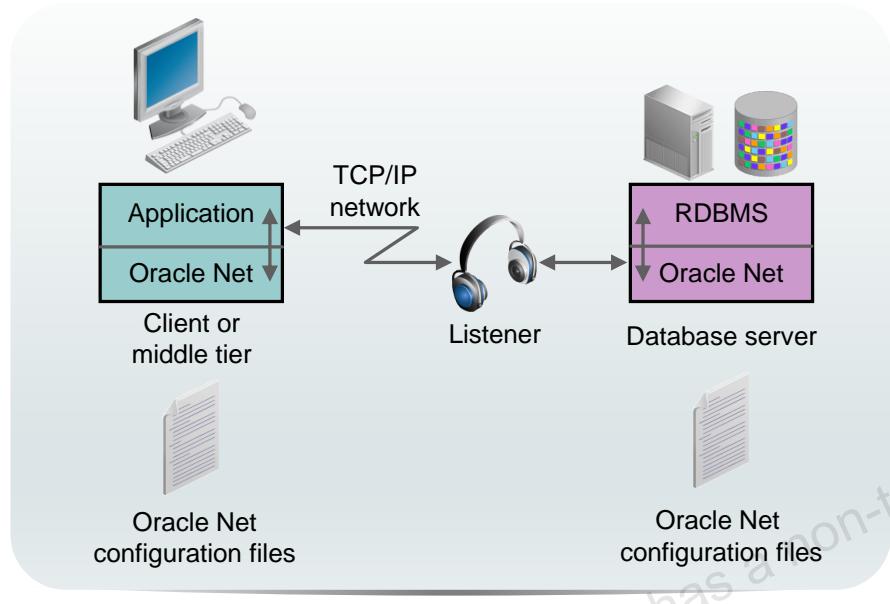


ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

This lesson takes a close look at the Oracle Net Listener and how you can configure your database instance to either dynamically or statically register services to the listener. In addition to the connection methods you learned in an earlier lesson (Easy Connect and OS authentication), you learn that with a little bit of setup, you can quickly connect to a database service by using the local naming method. Lastly, you become familiar with some of the more advanced networking concepts, such as shared server configurations.

Oracle Net Services: Overview



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

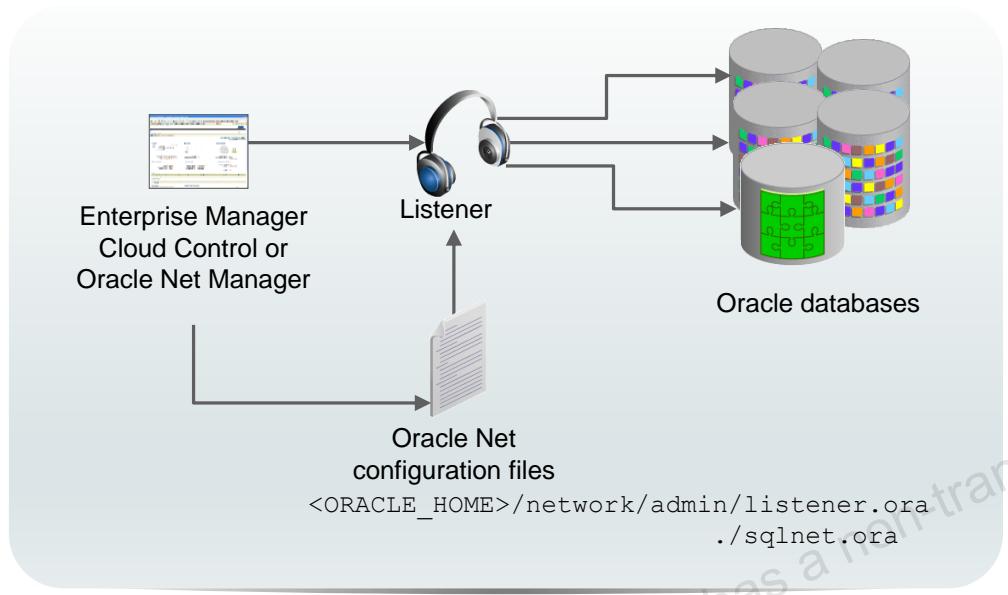
Oracle Net Services enables network connections from a client or middle-tier application to the Oracle server. After a network session is established, Oracle Net acts as the data courier for both the client application and the database server. It is responsible for establishing and maintaining the connection between the client application and database server, as well as exchanging messages between them. Oracle Net (or something that simulates Oracle Net, such as Java Database Connectivity) is located on each computer that needs to talk to the database server.

On the client computer, Oracle Net is a background component for application connections to the database.

On the database server, Oracle Net includes an active process called *Oracle Net Listener*, which is responsible for coordinating connections between the database and external applications.

The most common use of Oracle Net Services is to allow incoming database connections. You can configure additional net services to allow access to external code libraries (EXTPROC) and to connect the Oracle instance to non-Oracle data sources through Oracle Heterogeneous Services.

Oracle Net Listener: Overview



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Net Listener (or simply *the listener*) is the gateway to the Oracle instance for all nonlocal user connections. A single listener can service multiple database instances and thousands of client connections.

You can use Enterprise Manager Cloud Control or Oracle Net Manager to configure the listener and specify log file locations.

Advanced administrators can also configure Oracle Net Services by manually editing the configuration files, if necessary, with a standard operating system (OS) text editor such as `vi` or `gedit`.

The Default Listener

- During an Oracle Database installation, Oracle Universal Installer launches Oracle Net Configuration Assistant and creates a local listener named LISTENER.
- LISTENER is automatically populated with available database services through a feature called dynamic service registration.
- LISTENER listens on the following TCP/IP protocol address:
ADDRESS=(PROTOCOL=tcp) (HOST=host_name) (PORT=1521))
- Without any configuration, you can access your database instance immediately through LISTENER.
- If the listener name is LISTENER and it cannot be resolved, a protocol address of TCP/IP and a port number of 1521 is assumed.

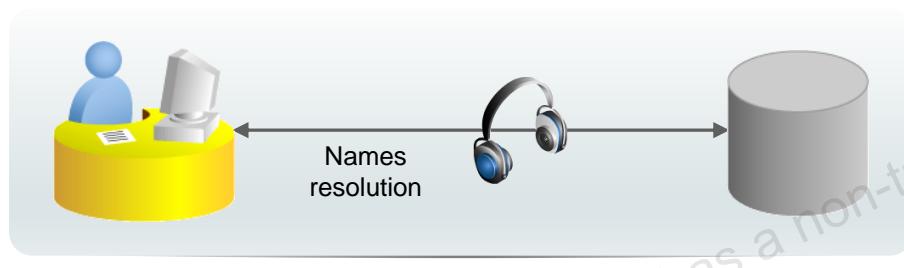


Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Establishing Oracle Network Connections

To make a client or middle-tier connection, Oracle Net requires the client to know the:

- Host where the listener is running
- Port that the listener is monitoring
- Protocol that the listener is using
- Name of the service that the listener is handling



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

For an application to connect to a service through Oracle Net Listener, it must have information about that service, including the address or host where the listener resides, the protocol that the listener accepts, and the port that the listener monitors. After the listener is located, the final piece of information that the application needs is the name of the service to which it wants to connect.

Oracle Net names resolution is the process of determining this connection information.

Connecting to an Oracle Database



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

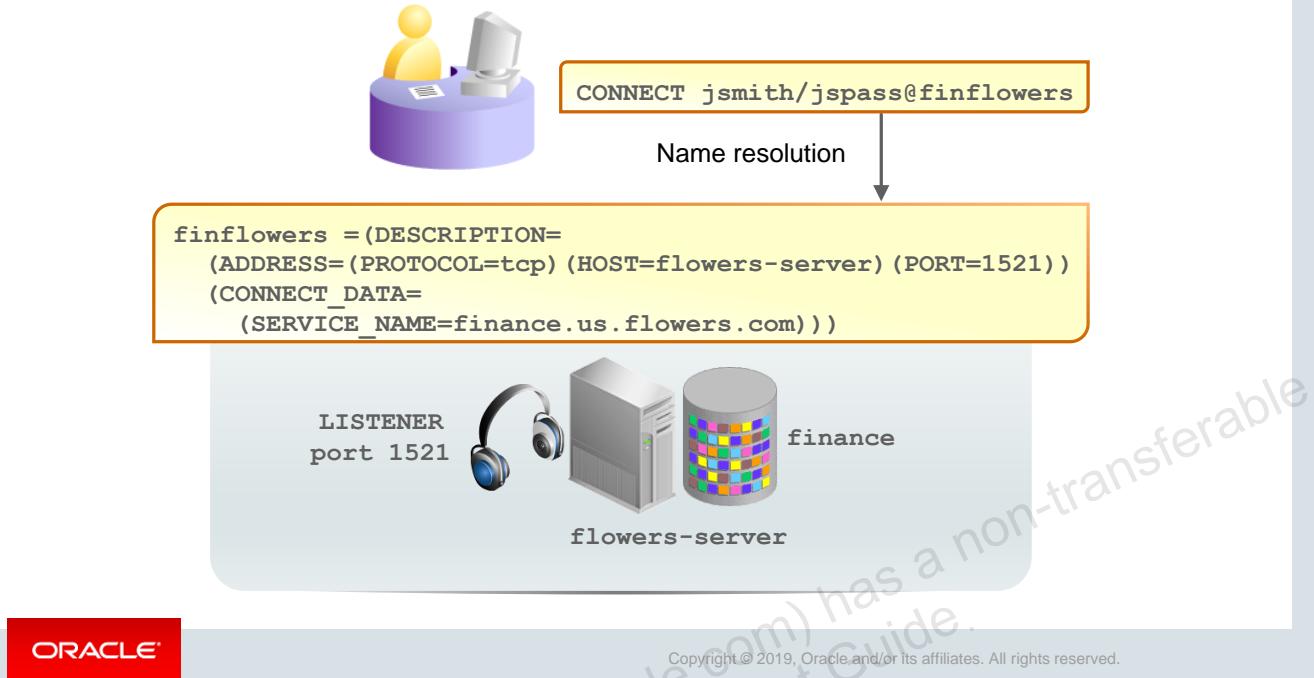
An Oracle database is represented to a client as a service. A database can have one or more services associated with it. Databases are identified by a *service name* that is specified by the `SERVICE_NAMES` parameter in the initialization parameter file. The service name defaults to the global database name, which is a name that comprises the database name (`DB_NAME` parameter value) and the domain name (`DB_DOMAIN` parameter value).

To connect to a database service, clients use a *connect descriptor* that provides the location of the database and the name of the database service. Clients can use the connect descriptor or a name that resolves to the connect descriptor (as discussed later in this lesson).

The following example shows a connect descriptor that enables clients to connect to a database service called `finance.us.flowers.com`.

```
(DESCRIPTION=
  (ADDRESS= (PROTOCOL=tcp) (HOST=flowers-server) (PORT=1521) )
  (CONNECT_DATA=
    (SERVICE_NAME=finance.us.flowers.com) ))
```

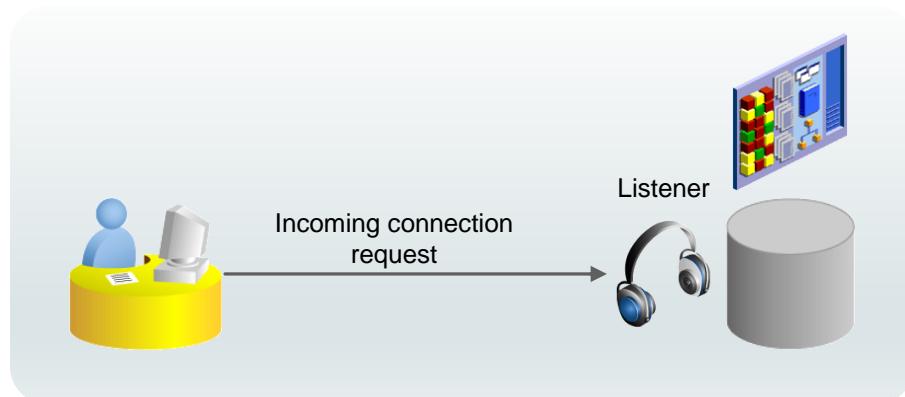
Name Resolution



Users initiate a connection request to the Oracle database by sending a *connect string*. A connect string includes a username and password, along with a *connect identifier*. A connect identifier can be the connect descriptor itself or a *name* that resolves to a connect descriptor. One of the most common connect identifiers is a *net service name*, which is a simple name for a service.

When a net service name is used, connection processing takes place by mapping the net service name to a connect descriptor. The mapping information can be stored in one or more repositories of information and is resolved by using a *naming method*.

Establishing a Connection



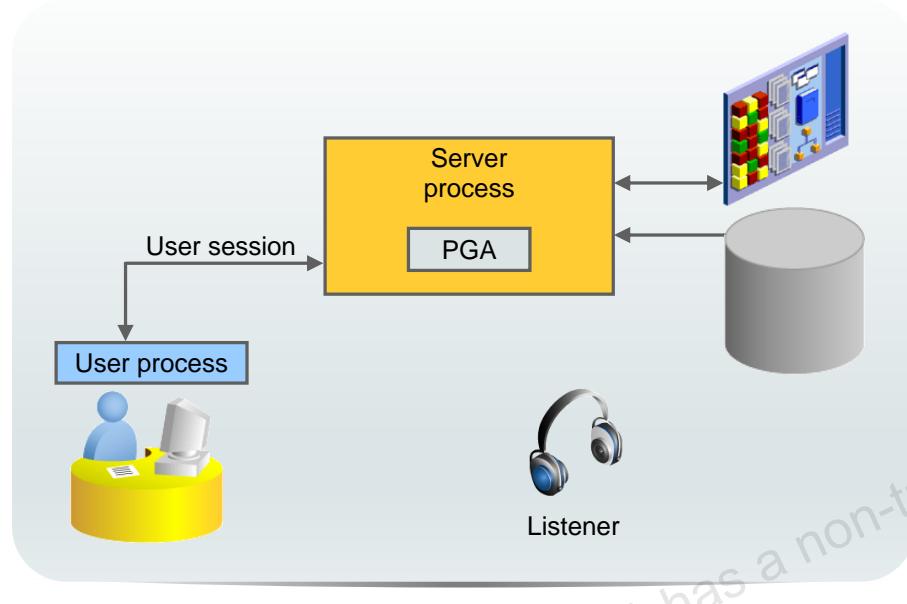
ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

After Oracle Net names resolution is complete, a connection request is passed from the user or middle-tier application (hereafter referred to as the *user process*) to the listener. The listener receives a CONNECT packet and checks whether that CONNECT packet is requesting a valid Oracle Net service name.

If the service name is not requested (as in the case of a tnsping request), the listener acknowledges the connect request and does nothing else. If an invalid service name is requested, the listener transmits an error code to the user process.

User Sessions



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

If the CONNECT packet requests a valid service name, the listener spawns a new process to deal with the connection. This new process is known as the *server process*. The listener connects to the process and passes the initialization information, including the address information for the user process. At this point, the listener no longer deals with the connection, and all work is passed to the server process.

The server process checks the user's authentication credentials (usually a password), and if the credentials are valid, a user session is created.

Dedicated server process: With the session established, the server process now acts as the user's agent on the server. The server process is responsible for:

- Parsing and running any SQL statements issued through the application
- Checking the database buffer cache for data blocks required to perform SQL statements
- Reading necessary data blocks from data files on the disk into the database buffer cache portion of the System Global Area (SGA), if the blocks are not already present in the SGA
- Managing all sorting activity. The Sort Area is a memory area that is used to work with sorting; it is contained in a portion of memory that is associated with the Program Global Area (PGA).
- Returning results to the user process in such a way that the application can process the information
- Reading auditing options and reporting user processes to the audit destination

Configuring Dynamic Service Registration

- By default, an Oracle database is configured to use dynamic service registration (service registration), which allows the Oracle database to identify its available services to listeners automatically.
- The LREG process polls the listeners to see if they're running and, if so, registers database service information to them.
- Dynamic service registration registers, by default, all PDB services to the same listener. If you stop that listener, you stop access to all the PDB services.
- General steps to configure dynamic service registration:
 - Make sure that the `INSTANCE_NAME`, `LOCAL_LISTENER`, `REMOTE_LISTENER`, and `SERVICE_NAMES` initialization parameters are properly configured.
 - Configure protocol addresses (end points) in the server-side `tnsnames.ora` file.



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Benefits of Dynamic Service Registration

Service registration offers the following benefits:

- Connect-time failover: Because the listener always monitors the state of the instances, service registration facilitates automatic failover of a client connect request to a different instance if one instance is down.
- Connection load balancing: Service registration enables the listener to forward client connect requests to the least-loaded instance and dispatcher or dedicated server. Service registration balances the load across the service handlers and nodes.
- High availability for Oracle Real Application Clusters and Oracle Data Guard

The Role of the LREG Process

The Listener Registration (LREG) process polls the listeners to see if they're running and, if so, registers the following database service information to them:

- Database instance name
- Database service names available on the database instance (for example, `ORCL.example.com` and `PDB1.example.com`)
- Current and maximum load for the database instance
- Service handlers (dispatchers and dedicated servers) available to the database instance

LREG registers with the listeners after the database instance mounts the database and every 60 seconds afterward.

How to Configure Dynamic Service Registration

The LREG process learns of the available listeners through the LOCAL_LISTENER and REMOTE_LISTENER parameters. These parameters specify listener alias names for local listeners and remote listeners. Both parameters can have multiple values. These aliases resolve to protocol addresses (end points) in the server-side tnsnames.ora file.

Note: Clients can also have a tnsnames.ora file, which you'll learn about in a later lesson.

Through dynamic service registration, the LREG process is then able to pass on information about the available database services to all listeners on those end points.

For example, assume LOCAL_LISTENER = LISTENER_HOST1 and REMOTE_LISTENER = LISTENER_HOST2. In the tnsnames.ora file, the LISTENER_HOST1 and LISTENER_HOST2 aliases are resolved to two different end points on two different machines. Notice that the CONNECT_DATA section is not included.

```
LISTENER_HOST1 =
  (ADDRESS = (PROTOCOL = TCP) (HOST = host1.example.com) (PORT = 1521))
LISTENER_HOST2 =
  (ADDRESS = (PROTOCOL = TCP) (HOST = host2.example.com) (PORT = 1521))
```

For dynamic service registration to work properly, make sure that the INSTANCE_NAME, LOCAL_LISTENER, REMOTE_LISTENER, and SERVICE_NAMES parameters are configured properly. By default, the installer populates the SERVICE_NAME parameter with the global database name (for example, ORCL.example.com), which provides one database service name that users can use to access the database instance. You can specify multiple service names for the database instance, however, if you want to distinguish among different uses of the same database. Oracle Database Resource Manager lets you view information about the user activity for each service name.

Dynamic service registration does not use the listener.ora file.

Configuring Static Service Registration

- Static service registration is a method for configuring listeners to obtain their service information manually.
 - You can create a listener for a particular PDB.
 - Static service registration might be required for some services, such as external procedures and heterogeneous services (for non-Oracle systems).
- With static registration, the listener has no knowledge of whether its database services exist or not. It only knows that it supports them. The Listener Configuration utility shows the services status as UNKNOWN.
- You can have both static listeners and dynamic listeners configured at the same time.
- General steps to configure static service registration:
 1. In `listener.ora`, define a listener and its protocol addresses.
 2. In `listener.ora`, also create a `SID_LIST_<listener name>` section that lists the database services for the listener.



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Advantages of Static Service Registration

The following are some advantages of using static service registration:

- Static service registration enables you to create a listener for a particular PDB.
- Sometimes you may need the database instance up and running without anyone being able to log in. As soon as it is started up, dynamic service registration will automatically start registering all the database services to the listener, making the database instance available to users.
- There is also a difference in error messages returned between a static listener (which can point to a database service that is down) and a dynamic listener entry (which shows nonexistence) when the database instance is shut down. The first case knows about the database service's existence and gives you an error message with useful information. The second case has no information and can't distinguish between a typo you may have made in the service name and whether it actually even exists.

Required Use of Static Service Registration

Static service registration is required for the following:

- Use of external procedure calls
- Use of Oracle Heterogeneous Services
- Use of Oracle Data Guard
- Remote database startup from a tool other than Oracle Enterprise Manager Cloud Control

How to Configure Static Service Registration

To create a static listener, you configure the `listener.ora` file. In that file, you define two sections. First, define the listener and its protocol addresses. Second, create a `SID_LIST_<listener name>` section that lists the database services for the listener. For each service, include the following parameters:

- `GLOBAL_DBNAME`: The PDB's service name (for example, `PDB1.example.com`)
- `ORACLE_HOME`: The Oracle home directory
- `SID_NAME`: The name of your database instance (for example, `ORCL`)

By default, the `listener.ora` file is stored in the `$ORACLE_HOME/network/admin` directory on the database instance machine.

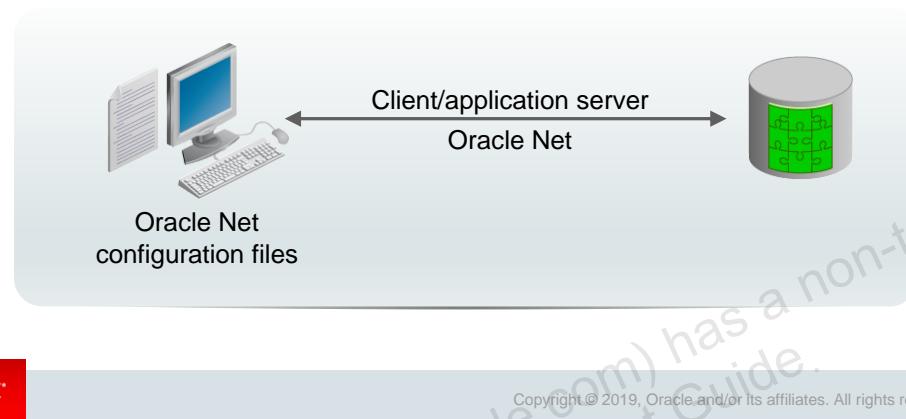
```
LISTENER_SALESPDBS =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP) (HOST = host1.example.com) (PORT = 1561))
    )
  )

SID_LIST_LISTENER_SALESPDBS =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = PDB1.example.com)
      (SID_NAME = ORCL)
      (ORACLE_HOME = /u01/app/oracle/product/12.2.0/dbhome_1)
    )
    (SID_DESC =
      (GLOBAL_DBNAME = PDB2.example.com)
      (SID_NAME = ORCL)
      (ORACLE_HOME = /u01/app/oracle/product/12.2.0/dbhome_1)
    )
  )
)
```

Naming Methods

Oracle Net supports several methods of resolving connection information:

- Easy connect naming: Uses a TCP/IP connect string
- Local naming: Uses a local configuration file
- Directory naming: Uses a centralized LDAP-compliant directory server



ORACLE®

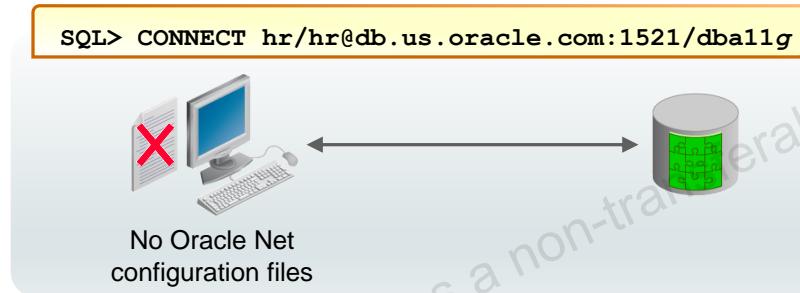
Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Net provides support for the following naming methods:

- **Easy connect naming:** The easy connect naming method enables clients to connect to an Oracle Database server by using a TCP/IP connect string consisting of a host name, optional port, and service name as follows:
`CONNECT username/password@host[:port] [/service_name]`
The easy connect naming method requires no configuration.
- **Local naming:** The local naming method stores connect descriptors (identified by their net service name) in a local configuration file named `tnsnames.ora` on the client.
- **Directory naming:** To access a database service, the directory naming method stores connect identifiers in a centralized directory server that is compliant with the Lightweight Directory Access Protocol (LDAP).
- **External naming:** The external naming method stores net service names in a supported non-Oracle naming service. Supported third-party services include:
 - Network Information Service (NIS) External Naming
 - Distributed Computing Environment (DCE) Cell Directory Services (CDS)

Easy Connect

- Is enabled by default
- Requires no client-side configuration
- Supports only TCP/IP (no SSL)
- Offers no support for advanced connection options such as:
 - Connect-time failover
 - Source routing
 - Load balancing



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

With Easy Connect, you supply all the information that is required for the Oracle Net connection as part of the connect string. Easy Connect connection strings take the following form:

```
<username>/<password>@<hostname>:<listener port>/<service name>
```

The listener port and service name are optional. If the listener port is not provided, Oracle Net assumes that the default port of 1521 is being used. If the service name is not provided, Oracle Net assumes that the database service name and host name provided in the connect string are identical.

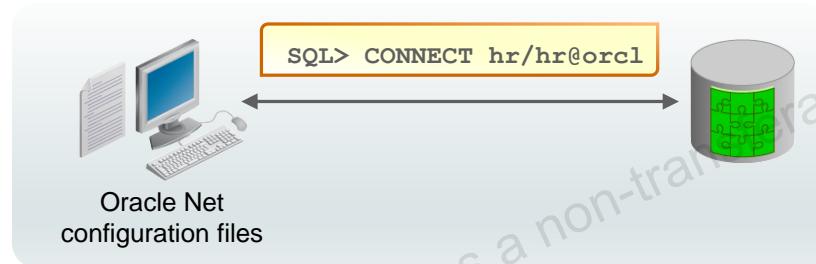
Assuming that the listener uses TCP to listen on port 1521, and the SERVICE_NAMES=db and DB_DOMAIN=us.oracle.com instance parameters, the connect string shown in the slide can be shortened:

```
SQL> connect hr/hr@db.us.oracle.com
```

Note: The SERVICE_NAMES initialization parameter can accept multiple comma-separated values. Only one of those values must be db for this scenario to work.

Local Naming

- Requires a client-side names-resolution file
- Supports all Oracle Net protocols
- Supports advanced connection options such as:
 - Connect-time failover
 - Source routing
 - Load balancing



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

With local naming, the user supplies an alias for the Oracle Net service. Oracle Net checks the alias against a local list of known services and, if it finds a match, converts the alias into host, protocol, port, and service name.

One advantage of local naming is that the database users need to remember only a short alias rather than the long connect string required by Easy Connect.

The local list of known services is stored in the following text configuration file:

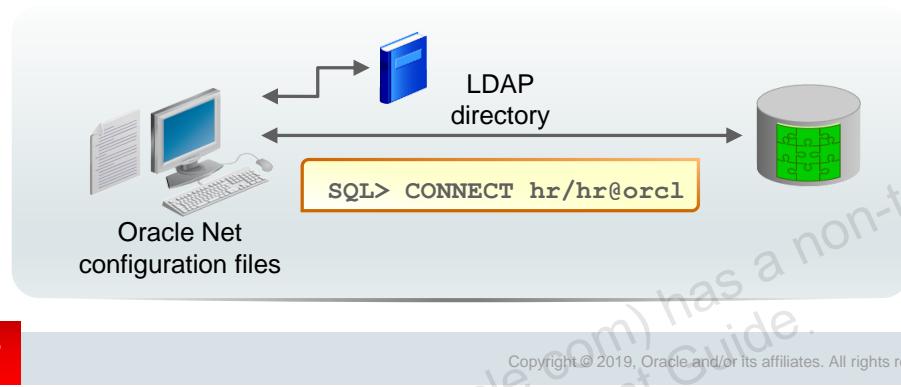
<oracle_home>/network/admin/tnsnames.ora

This is the default location of the `tnsnames.ora` file, but the file can be located elsewhere by using the `TNS_ADMIN` environment variable.

Local naming is appropriate for organizations in which Oracle Net service configurations do not change often.

Directory Naming

- Requires LDAP with Oracle Net names resolution information loaded:
 - Oracle Internet Directory
 - Microsoft Active Directory Services
- Supports all Oracle Net protocols
- Supports advanced connection options



With directory naming, the user supplies an alias for the Oracle Net service. Oracle Net checks the alias against an external list of known services and, if it finds a match, converts the alias into host, protocol, port, and service name. Like local naming, database users need to remember only a short alias.

One advantage of directory naming is that the service name is available for users to connect with as soon as a new service name is added to the LDAP directory. With local naming, the database administrator (DBA) must first distribute updated `tnsnames.ora` files containing the changed service name information before users can connect to new or modified services.

Directory naming is appropriate for organizations in which Oracle Net service configurations change frequently.

Tools for Configuring and Managing Oracle Net Services

- Enterprise Manager Net Services Administration page
- Oracle Net Manager
- Oracle Net Configuration Assistant
- Listener Control Utility



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Use the following tools and applications to manage your Oracle Network configuration:

- **Enterprise Manager Cloud Control:** Provides an integrated environment for configuring and managing Oracle Net Services. Use Enterprise Manager to configure Oracle Net Services for any Oracle home across multiple file systems and administer listeners.
- **Oracle Net Manager:** Provides a graphical user interface (GUI) through which you can configure Oracle Net Services for an Oracle home on a local client or a server host
- **Oracle Net Configuration Assistant:** Launched by Oracle Universal Installer when you install the Oracle software. During a typical database installation, Oracle Net Configuration Assistant automatically configures a listener called LISTENER that has a TCP/IP listening protocol address for the database. If you perform a custom installation, Oracle Net Configuration Assistant prompts you to configure a listener name and protocol address of your choice.
- **Listener Control Utility:** Used to start, stop, and view the status of the listener process

Defining Oracle Net Services Components

Component	Description	File
Listeners	A process that resides on the server whose responsibility is to listen for incoming client connection requests and manage traffic to the server.	listener.ora
Naming methods	A resolution method used by a client application to resolve a connect identifier to a connect descriptor when attempting to connect to a database service.	
Naming (net service name)	A simple name (connect identifier) for a service that resolves to a connect descriptor to identify the network location and identification of a service.	tnsnames.ora (local configuration)
Profiles	A collection of parameters that specifies preferences for enabling and configuring Oracle Net features on the client or server.	sqlnet.ora



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

The following Oracle Net Services components can be configured by using Enterprise Manager Cloud Control and Oracle Net Manager:

- Listener: Configuration of the listener includes specifying the listener name, protocol addresses it is accepting connection requests on, and services (database or nondatabase service) it is listening for.
- Naming (net service name)
- Naming methods
- Profiles

The Oracle Net Configuration Assistant configures the listener, naming methods, directory server usage, and a local `tnsnames.ora` file during installation of Oracle Database software.

Advanced Connection Options

- When a database service is accessible by multiple listener protocol addresses, you can specify the order in which the addresses are to be used.
- Oracle Net supports the following advanced connection options with local and directory naming:
 - Connect-time failover
 - Load balancing
 - Source routing



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

When a database service is accessible by multiple listener protocol addresses, you can specify the order in which the addresses are to be used. The addresses can be chosen randomly or tried sequentially. In cases in which more than one listener is available, such as Oracle Real Application Clusters (RAC) configurations, Oracle Net can take advantage of listener failover and load balancing as well as Oracle Connection Manager source routing.

Oracle Net supports advanced connection options, typically used in an Oracle RAC or Oracle Data Guard configuration:

- **Connect-Time Failover:** With connect-time failover enabled, the alias has two or more listener addresses listed. If the first address is not available, the second is tried. Oracle Net keeps trying addresses in the listed order until it reaches a listener that is functioning or until all addresses have been tried and failed. Transparent Application Failover (TAF) is a client-side feature that allows clients to reconnect to surviving databases in the event of a database instance failure. Notifications are used by the server to trigger TAF callbacks on the client side.
- **Load Balancing:** With load balancing enabled, Oracle Net picks an address at random from the list of addresses. The runtime connection load-balancing feature improves connection performance by balancing the number of active connections among multiple dispatchers. In a RAC environment, connection pool load balancing also has the capability to balance the number of active connections among multiple instances.
- **Source Routing:** Source routing is used with Oracle Connection Manager, which serves as a proxy server for Oracle Net traffic, enabling Oracle Net traffic to be routed securely through a firewall. Oracle Net treats the addresses as a list of relays, connecting to the first address and then requesting to be passed from the first to the second until the destination is reached. It differs from failover or load balancing in that all addresses are used each time a connection is made.

Testing Oracle Net Connectivity with tnsping

- The tnsping utility tests Oracle Net service aliases.
- It validates connectivity between a client and the Oracle Net Listener.
 - It validates that the host name, port, and protocol reach a listener.
 - It does not check whether the listener handles the service name.
 - It does not verify that the requested service is available.
- The tnsping utility also reveals the location of the configuration files. In a system with multiple ORACLE_HOME locations, this can be helpful.
- Examples:
 - tnsping supports Easy Connect names resolution:
\$tnsping host01.example.com:1521/orcl
 - tnsping also supports local and directory naming:
\$ tnsping orcl



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Configuring Communication Between Database Instances

- You must configure network connectivity (for example, `tnsnames.ora`) and a database link for database instance to database instance communication.
- A database link is a schema object that enables you to access objects on a different database.
- SQL command to create a fixed user, private database link:

```
CREATE DATABASE LINK <database_link_name>
CONNECT TO <user> IDENTIFIED BY <pwd>
USING '<connect_string_for_remote_db>'
```

- SQL command to query a table by using the database link:

```
SELECT * FROM employees@<database_link_name>
```



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Database-to-Database Communication

To configure communication from your database to another database, you must configure the following on your database server:

- Network connectivity (for example, a net service name defined in the server-side `tnsnames.ora` file)
- A database link

A database link is a schema object that enables you to access objects on a different database.

- The other database need not be an Oracle database system.
- To access non-Oracle systems, you must use Oracle Heterogeneous Services.

When an application uses a database link to access a remote database, Oracle Database establishes a database session in the remote database on behalf of the local request.

You can create fixed user, current user, and connected user database links. Current user links are available only through the Oracle Advanced Security option.

You can include the `PUBLIC` keyword to create a public database link that is visible to all users. If you omit `PUBLIC`, the database link is private and available only to you.

To create a private database link, you must have the `CREATE DATABASE LINK` system privilege. To create a public database link, you must have the `CREATE PUBLIC DATABASE LINK` system privilege.

Creating Database Links

The SQL command to create a database link is as follows. The CONNECT TO clause determines how the connection is established on the remote database. The <connect_string_for_remote_db> can be a net service name.

```
CREATE DATABASE LINK <database_link_name>
CONNECT TO <user> IDENTIFIED BY <pwd>
USING '<connect_string_for_remote_db>'
```

After you create a database link, you can use it to refer to tables and views on the other database. In SQL statements, you can refer to a table or view on the other database by appending @dblink to the table or view name. You can query a table or view on the other database or use any INSERT, UPDATE, DELETE, or LOCK TABLE statement for the table. For example:

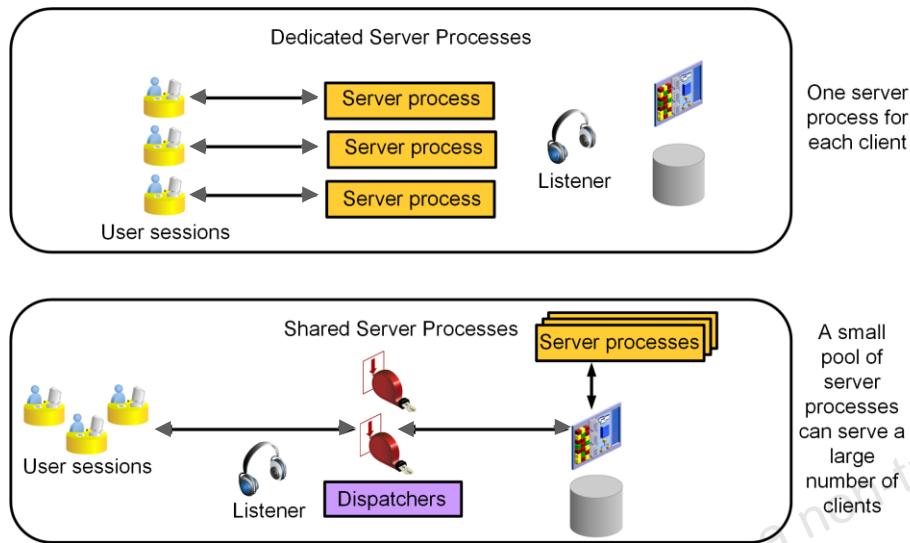
```
SELECT * FROM employees@<database_link_name>
```

Listing Database Links

You can query the following views to determine which database links are defined in your database:

- DBA_DB_LINKS: Lists all database links in the database
- ALL_DB_LINKS: Lists all database links accessible to the connected user
- USER_DB_LINKS: Lists all database links owned by the connected user

Comparing Dedicated and Shared Server Configurations



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Dedicated Server Configuration

In a dedicated server configuration, as illustrated in the slide, one server process handles requests for a single client process. Each server process uses system resources, including CPU cycles and memory. In a heavily loaded system, the memory and CPU resources that are used by dedicated server processes can be prohibitive and negatively affect the system's scalability. If your system is being negatively affected by the resource demands of the dedicated server architecture, you have the following options:

- Increase system resources by adding more memory and additional CPU capability
- Use the Oracle Shared Server Process architecture

Shared Server Configuration

A shared server configuration, as illustrated in the slide, enables multiple client processes to share a small number of server processes. Each service that participates in the shared server process architecture has at least one dispatcher process (and usually more). When a connection request arrives, the listener does not spawn a dedicated server process. Instead, the listener maintains a list of dispatchers that are available for each service name, along with the connection load (number of concurrent connections) for each dispatcher. Connection requests are routed to the lightest loaded dispatcher that is servicing a given service name. Users remain connected to the same dispatcher for the duration of a session.

Unlike dedicated server processes, a single dispatcher can manage hundreds of user sessions. Dispatchers do not actually handle the work of user requests. Instead, they pass user requests to a common queue located in the shared pool portion of the SGA. Shared server processes take over most of the work of dedicated server processes, pulling requests from the queue and processing them until they are complete.

Because a user session may have requests processed by multiple shared server processes, most of the memory structures that are usually stored in the PGA must be in a shared memory location (by default, in the shared pool). However, if the large pool is configured or if `SGA_TARGET` is set for automatic memory management, these memory structures are stored in the large pool portion of the SGA.

The contents of the SGA and PGA are different in a shared server configuration than it is in a dedicated server configuration. In a shared server configuration:

- Text and parsed forms of all SQL statements are stored in the SGA.
- The cursor state contains runtime memory values for the SQL statement, such as rows retrieved.
- User-session data includes security and resource usage information.
- The stack space contains local variables for the process.

The change in the SGA and PGA is transparent to the user; however, if you are supporting multiple users, you need to increase the `LARGE_POOL_SIZE` initialization parameter. Each shared server process must access the data spaces of all sessions so that any server can handle requests from any session. Space is allocated in the SGA for each session's data space. You limit the amount of space that a session can allocate by setting the `PRIVATE_SGA` resource.

Pros and Cons

A dedicated server process is good for long-running queries and administrative tasks and is faster than a shared server process in that there is always a server process ready to do work. However, an idle process or too many dedicated processes can result in an inefficient use of resources. Using shared server mode on the database server eliminates the need for a dedicated server process for each user connection, requires less memory for each user connection, and enables a larger number of users on a system with constrained memory. Dedicated server processes and shared server processes are enabled at the same time. Oracle XML DB (XDB) requires shared server processes, and the Oracle database is already configured to use them. You will need to modify the initialization parameters for other users to use shared server processes.

The Oracle Shared Server architecture is an efficient process and memory use model, but it is not appropriate for all connections. Because of the common request queue and the fact that many users may share a dispatcher response queue, shared servers do not perform well with operations that must deal with large sets of data, such as warehouse queries or batch processing. Backup and recovery sessions that use Oracle Recovery Manager (discussed in later lessons) also deal with very large data sets and must use dedicated connections. Many administration tasks must not (and cannot) be performed by using shared server connections. These include starting up and shutting down the instance, creating tablespaces and data files, maintaining indexes and tables, analyzing statistics, and many other tasks that are commonly performed by the DBA. All DBA sessions must choose dedicated servers.

Summary

In this lesson, you should have learned how to:

- Describe Oracle Net Services
- Explain how listeners work
- Configure listeners for dynamic or static service registration
- Configure local naming for database connections
- Test Oracle Net connectivity with `tnsping`
- Configure communication between databases by creating database links
- Explain the difference between dedicated and shared server configurations



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Practice 8: Overview

- 8-1: Exploring the Default Listener
- 8-2: Creating a Static Listener for a PDB
- 8-3: Verifying the Net Service Name for MYPDB1



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Administering User Security

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

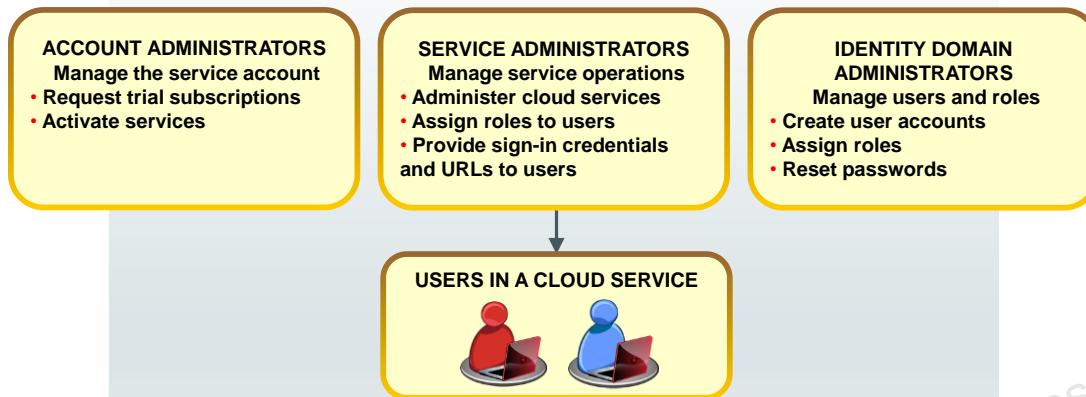
- Create database users
- Grant privileges to database users
- Create and grant roles to users or other roles
- Revoke privileges and roles from users and other roles
- Create and assign profiles to users
- Explain the various authentication options for users
- Assign quota to users
- Apply the principle of least privilege



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Cloud User Roles and Privileges



See [Oracle Cloud User Roles and Privileges](#) in *Getting Started with Oracle Cloud* for additional information.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If you have subscribed to an entitlement to create instances of an Oracle Cloud Service, then you can create multiple service instances based on your business needs.

You have identity domain administrator privileges to create a user and assign the service entitlement administrator role to the user, who will create service instances. The actual name of this role depends on the Oracle Cloud service that you have subscribed to.

Identity domain administrators can assign and remove roles only for the users in the identity domains that they manage.

Service administrators (or the DBCS database administrator) can assign and remove roles only for the users of the services that they manage. Because service administrators cannot add users or roles, the users and roles must already be in the system before service administrators can assign a specific role to a user.

Administering Oracle Cloud Users, Roles, and Privileges



- Administer Cloud Services users by accessing the Users page.
- Cloud Services users are different from Oracle Database users.

The screenshot shows the Oracle Cloud My Services interface. In the top left, it says "ORACLE CLOUD My Services". In the top right, there are "Dashboard" and "Users" buttons. The "Users" button is highlighted. Below the header, it says "Data Center: EMEA Commercial 2 - Amsterdam (Time zone: Europe/Amsterdam)". Under the "Users" tab, there's a sub-header "Manage user accounts, assign roles, and reset passwords." A search bar with "Find user" and a magnifying glass icon is present. To the right of the search bar are buttons for "Show: All Roles" and "Sort by: First Name". At the bottom of the list, there's a small user icon and the name "Db Dev" followed by the email and user name. At the very bottom of the page, there's a red "ORACLE" logo and a copyright notice: "Copyright © 2019, Oracle and/or its affiliates. All rights reserved."

See [Creating a User and Assigning a Role in Getting Started with Oracle Cloud](#) for details.

The Users page may be selected from various consoles. On the Users page, you can add individual users and configure the roles (privileges) for each user.

User privileges are controlled through assigned roles. The roles assign privileges in a granular fashion, and each role has specific privileges.

The Import button allows you to import a list of users as a CSV file (comma delimited).

The Users page provides administration for:

- **SFTP users for file transfers:** Predefined users with specific uses
- **Roles:** Predefined roles for managing cloud services; roles are assigned to individual users, or a single role can be assigned to a list of users in a CSV file
- **Custom Roles:** Created and deleted by identity domain administrators and then used by application developers to secure applications
- **SSO Configuration:** Used to configure Single Sign-On so your users can use their company credentials to log in to all applications, including Oracle Cloud applications
- **OAuth Administration:** Used to manage client access to Oracle Cloud APIs using the OAuth 2.0 protocol
- **My Profile:** Each user can modify basic user information, reset the password, change the roles, and remove the user.

For each user, there is also a menu icon that can be used to modify basic user information, reset the password, change the roles, and remove the user.

Managing Oracle Cloud Compute Node Users



When a database deployment is created, three Linux users are created.

OS User	Authorization
opc	Authorized to log in to the compute node Authorized to run <code>root</code> commands Can use <code>sudo -s</code>
oracle	Authorized to log in to the compute node Not authorized to run <code>root</code> commands
root	Not authorized to log in to the compute node

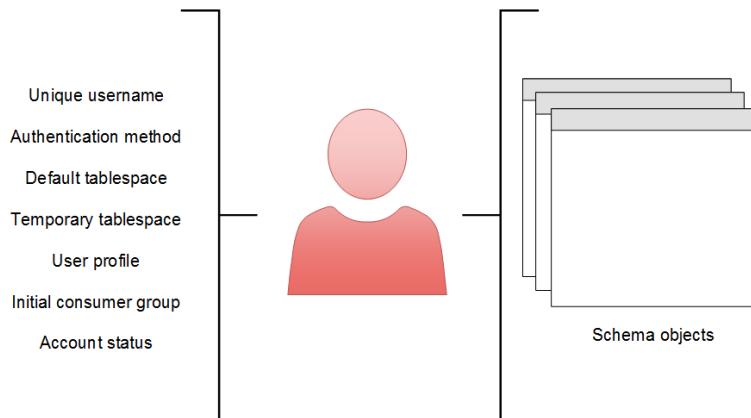


Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When a database deployment is created, three Linux users are created:

- `opc`: The `opc` user is authorized to run `root` commands. Certain operations such as database backup and recovery or compute node reboot require the use of the `sudo` command. Connect as `opc` to perform these operations.
- `oracle`: The `oracle` user is the administrator account you use to access the system and perform operations on the compute node. A home directory, `/home/oracle`, is created for this user. This user cannot use the `sudo` command to perform operations that require root-user access.
- `root`: The `root` administrator for the system exists, but you do not have a direct connection to this account. To perform operations that require root-user access, connect as the `opc` user and use the `sudo` command.

Database User Accounts



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

User Account Components

To access the database, a user must specify a valid database user account and successfully authenticate as required by that user account. Each database user has a unique database account. Oracle recommends this to avoid potential security holes and provide meaningful data for certain audit activities. However, users may sometimes share a common database account. In these rare cases, the operating system and applications must provide adequate security for the database.

Each user account has the following, as illustrated in the slide:

- **Unique username:** Usernames cannot exceed 30 bytes, cannot contain special characters, and must start with a letter.
- **Authentication method:** The most common authentication method is a password.
- **Default tablespace:** This is a place where a user creates objects if the user does not specify some other tablespace.
 - Having a default tablespace does not imply that the user has the privilege of creating objects, nor does the user have a quota of space in that tablespace in which to create objects. Both of these privileges are granted separately.
 - If a user does not specify a tablespace when creating an object, the object will be created in the default tablespace assigned to the object owner. This enables you to control where the user's objects are created.
 - If an administrator does not define a default tablespace, the system-defined default permanent tablespace is used.

- Quota for a specific `tablespace` is not granted through a privilege. It's done by using the `ALTER USER` command, which changes the attributes for a user. However, if a DBA grants the `UNLIMITED TABLESPACE` system privilege to a user, then that user can use all the space in any tablespace.
- **Temporary tablespace:** This is a place where temporary objects, such as sorts and temporary tables, are created on behalf of the user by the instance. No quota is applied to temporary tablespaces. If an administrator does not define a temporary tablespace for a user, the system-defined temporary tablespace is used when the user creates objects.
- **User profile:** This is a set of resource and password restrictions assigned to the user.
- **Initial consumer group:** This is used by the Resource Manager.
- **Account status:** Users can access only “open” accounts. The account status may be “locked” and/or “expired.”

Note: A database user is not necessarily a person. It is a common practice to create a user that owns the database objects of a particular application, such as `HR`. The database user can be a device, an application, or just a way to group database objects for security purposes. The personal identifying information of a person is not needed for a database user.

Schemas

A schema is a collection of database objects that are owned by a database user. Schema objects are the logical structures that directly refer to the database’s data. Schema objects include such structures as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. In general, schema objects include everything that your application creates in the database.

Oracle-Supplied Administrator Accounts

Account	Description
SYS	Super user. Owns the data dictionary and the Automatic Workload Repository (AWR). Used for starting up and shutting down the database instance.
SYSTEM	Owns additional administrative tables and views
SYSBACKUP	Facilitates Oracle Recovery Manager (RMAN) backup and recovery operations
SYSDG	Facilitates Oracle Data Guard operations
SYSKM	Facilitates Transparent Data Encryption wallet operations
SYSRAC	For Oracle Real Application Clusters (RAC) database administration tasks
SYSMAN	For Oracle Enterprise Manager database administration tasks
DBSNMP	Used by the Management Agent component of Oracle Enterprise Manager to monitor and manage the database

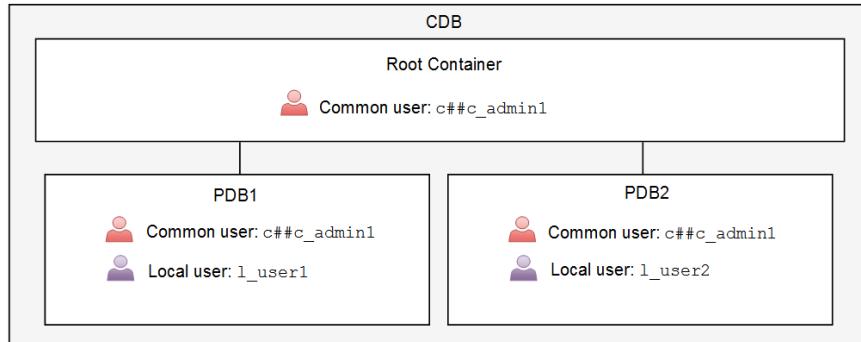


Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The SYS and SYSTEM accounts are required accounts in the database and cannot be deleted. You supply their passwords when you create the database instance and database in DBCA.

During installation and database creation, you can unlock and reset many of the Oracle-supplied database user accounts.

Creating Oracle Database Users in a Multitenant Environment



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Comparing Common and Local Users

A multitenant environment has two types of database users that you can create:

- **Common user:** The user is replicated in all existing and future containers. Oracle supplies several common user accounts for database administrators to use. A common user that you create, by default, must be given a name that starts with C## or c## (for example, c##c_admin1). The COMMON_USER_PREFIX parameter specifies a prefix for common users, roles, and profiles in a container database. To create a common user, log in to the root container, issue the CREATE USER command, and include the CONTAINER=ALL clause. For example:

```
SQL> CONNECT / AS SYSDBA
SQL> CREATE USER c##c_admin1 IDENTIFIED BY x CONTAINER=ALL;
```

- **Local user:** The user is created in a single PDB only. Local users cannot be created in a root container or in an application root container. A local user cannot create a common user. To create a local user, log in to the PDB where you want to create the local user and issue the CREATE USER command. For example:

```
SQL> CONNECT SYS@PDB1 AS SYSDBA
SQL> CREATE USER l_user1 ... ;
```

In the illustration in the slide, the common user c##c_admin1 exists in the root container and all PDBs, while local user l_user1 only exists in PDB1 and local user l_user2 only exists in PDB2.

You can use tools such as SQL*Plus and Enterprise Manager Database Express to create user accounts in the Oracle database.

Schema-Only Account

Ensure that a user cannot log in to the instance:

- Enforce data access through the application.
- Secure schema objects.
 - Prevent objects from being dropped by the connected schema.
- Use the NO AUTHENTICATION clause.
 - Can be replaced by IDENTIFIED BY VALUES
- A schema-only account cannot be:
 - Granted system administrative privileges
 - Used in database links

```
DBA_USERS  
AUTHENTICATION_TYPE = NONE | PASSWORD
```

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Application designers may want to create accounts that contain the application data dictionary, but are not allowed to log in to the instance. This can be used to enforce data access through the application, separation of duties at the application level, and other security mechanisms.

In addition, utility accounts can be created but remain inaccessible by denying the ability to log in except under controlled situations.

Until Oracle Database 12c, DBAs create accounts that do not need to log in to the instance or maybe rarely log in to the instance. Nevertheless, for all these accounts, there are default passwords and requirements to rotate the passwords.

In Oracle Database 18c, an account can be created with the NO AUTHENTICATION clause to ensure that the account is not permitted to log in to the instance. Removing the password and the ability to log in essentially just leaves a schema. The schema account can be altered to allow login, but can then have the password removed. The ALTER USER statement can be used to disable or re-enable the login capability.

The DBA_USERS view has a new column, AUTHENTICATION_TYPE, which displays NONE when NO AUTHENTICATION is set, and PASSWORD when a password is set.

Authenticating Users

- Every user, including administrators, must be authenticated when connecting to a database instance.
- Authentication verifies that the user is a valid database user and establishes a trust relationship for further interactions.
- Authentication also enables accountability by making it possible to link access and actions to specific identities.
- The following authentication methods are possible:
 - Password (usually for database users)
 - Operating system (OS) authentication
 - Password file (for system administrative privileged users only)
 - Strong authentication with Kerberos, SSL, or directory authentication
- A system administrative privileged user must use OS authentication, password file authentication, or strong authentication. These methods can authenticate when the database is available or unavailable (not started).



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Your choice of authentication is influenced by whether you intend to administer your database locally on the same system where the database resides or whether you intend to administer many different databases from a single remote client.

Password Authentication

- Password authentication is also referred to as "authentication" by the Oracle Database server.
- Create each user with an associated password that must be supplied when the user attempts to establish a connection.
- When setting up a password, you can expire the password immediately, which forces the user to change the password after first logging in.
 - If you decide on expiring user passwords, make sure that users have the ability to change the password. Some applications do not have this functionality.
 - All passwords created in Oracle Database are case-sensitive by default.
 - Passwords may contain multibyte characters and are limited to 30 bytes.
- Passwords are always automatically and transparently encrypted by using the Advanced Encryption Standard (AES) algorithm during network (client/server and server/server) connections before sending them across the network.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Password File Authentication

- You can use password file authentication for an Oracle database instance and for an Oracle Automatic Storage Management (Oracle ASM) instance.
- If authentication succeeds, the connection is logged with the `SYS` user.
- A password file stores database usernames and case-sensitive passwords for administrator users (common and local administrators).
- DBCA creates a password file during installation.
- To prepare for password file authentication, you must:
 - Create the password file.
 - Set the `REMOTE_LOGIN_PASSWORDFILE` initialization parameter.
 - Grant system administrative privileges (for example, `GRANT SYSDBA TO mydba`).
- Use the `CONNECT` command in SQL*Plus to connect. For example:

```
SQL> CONNECT mydba AS SYSDBA
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

For more information, see the following sources in *Oracle Database Administrator's Guide*:

- Preparing to Use Password File Authentication
- Connecting Using Password File Authentication

On UNIX and Linux, the password file is called `orapwORACLE_SID` and is stored in `$ORACLE_HOME/dbs`. On Windows, the file is called `PWDORACLE_SID.ora` and is stored in `$ORACLE_HOME\database`.

If your concern is that the password file might be vulnerable or that the maintenance of many password files is a burden, strong authentication can be implemented. You can query `V$PWFILE_USERS` to view information in the password file.

OS Authentication

- Oracle Universal Installer creates operating system groups, assigns them specific names, and maps each group to a specific system privilege.
 - Example: Members of the dba group are granted SYSDBA
- As a group member, you can be authenticated, enabled as an administrative user, and connected to a local database:


```
SQL> CONNECT / AS SYSDBA
SQL> CONNECT / AS SYSOPER
SQL> CONNECT / AS SYSBACKUP
SQL> CONNECT / AS SYSDG
SQL> CONNECT / AS SYSKM
SQL> CONNECT / AS SYSRAC
```
- If you are not a member of one of these OS groups, you will not be able to connect as an administrative user via OS authentication.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

OS Authentication for Privileged Users

Oracle Universal Installer creates operating system groups, assigns them specific names, and maps each group to a specific system privilege. The table in the next slide shows this mapping for a UNIX or Linux environment. Membership in one of these operating system groups enables a database administrator to authenticate to the database instance through the operating system rather than with a database username and password. This is known as operating system authentication.

Example

The special system privileges are not exercised unless you include them in your CONNECT clause. For example, assume that the HR user is granted the SYSDBA privilege and connects with that privilege. Notice that the current user becomes SYS:

```
SQL> CONNECT hr/hr@PDB1 AS SYSDBA
Connected.
SQL> SHOW USER
USER is "SYS"
```

However, if the HR user logs in to PDB1 without including the AS SYSDBA clause, the current user is HR and the user does not have the SYSDBA privilege.

```
SQL> CONNECT hr/hr@PDB1
Connected.
SQL> SHOW USER
USER is "HR"
```

OS Authentication for Users

If your operating system permits, you can have it authenticate users. They will not need to provide a username or password when connecting to the database instance.

If you use operating system authentication, set the `OS_AUTHENT_PREFIX` initialization parameter and use this prefix in Oracle usernames. The `OS_AUTHENT_PREFIX` parameter defines a prefix that the Oracle database adds to the beginning of each user's operating system account name. The default value of this parameter is `OPS$` for backward compatibility with the previous versions of the Oracle software. The Oracle database compares the prefixed username with the Oracle usernames in the database when a user attempts to connect. For example, assume that `OS_AUTHENT_PREFIX` is set as follows:

```
OS_AUTHENT_PREFIX=OPS$
```

If a user with an operating system account named `tsmith` needs to connect to an Oracle database and be authenticated by the operating system, the Oracle database checks whether there is a corresponding database user `OPS$tsmith` and, if so, allows the user to connect. All references to a user who is authenticated by the operating system must include the prefix, as seen in `OPS$tsmith`. The text of the `OS_AUTHENT_PREFIX` initialization parameter is case-sensitive on some operating systems.

OS Authentication for Privileged Users

OS Group	UNIX or Linux User Group	Special System Privilege Granted to Members
Oracle Software Group (top level group)	oinstall	Allowed to create and delete database files on the OS. All database administrators belong to this group.
Database Administrator Group (OSDBA)	dba	SYSDBA (Connects you as the SYS user)
Database Operator Group (OSOPER) – optional	oper	SYSOPER (Connects you as the PUBLIC user)
Database Backup and Recovery Group (OSBACKUPDBA)	backupdba	SYSBACKUP
Data Guard Administrative Group (OSDGDBA)	dgdba	SYSDG
Encryption Key Management Administrative Group (OSKMDBA)	kmdba	SYSKM
Real Application Cluster Administrative Group (OSRACDBA)	rac	SYSRAC



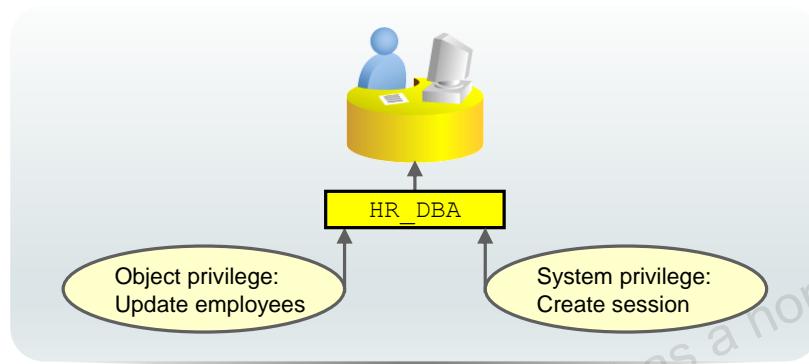
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If you are not a member of one of the OS groups listed in the slide, you will not be able to connect as an administrative user via OS authentication. That is, CONNECT / AS SYSDBA will fail. However, you can still connect using other authentication methods (for example, network, password, or directory-based authentication).

Privileges

There are two types of user privileges:

- System: Enables users to perform particular actions in the database
- Object: Enables users to access and manipulate a specific object



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

A **privilege** is a right to execute a particular type of SQL statement or to access another user's object.

Privileges are divided into two categories:

- **System privileges:** Each system privilege allows a user to perform a particular database operation or class of database operations. For example, the privilege to create tablespaces is a system privilege. System privileges can be granted by the administrator or by someone who has been given explicit permission to administer the privilege. There are more than 170 distinct system privileges. Many system privileges contain the ANY clause.
- **Object privileges:** Object privileges allow a user to perform a particular action on a specific object, such as a table, view, sequence, procedure, function, or package. Without specific permission, users can access only their own objects. Object privileges can be granted by the owner of an object, by the administrator, or by someone who has been explicitly given permission to grant privileges on the object.

System Privileges

- Each system privilege allows a user to perform a particular database operation or class of database operations.
- Administrators have special system privileges.
- A system privilege with the ANY clause means the privilege applies to all schemas, not just your own.
- If you grant a system privilege with the ADMIN OPTION enabled, you enable the grantee to administer the system privilege and grant it to other users.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

System Privileges

Each system privilege allows a user to perform a particular database operation or class of database operations. System privileges can be granted by the administrator or by someone who has been given explicit permission to administer the privilege. You can administer system privileges when you create a user or at a later time. Carefully consider security requirements before granting system permissions.

There are more than 170 distinct system privileges. A few are listed here:

- CREATE SESSION: Enables a user to connect to a database instance
- DROP ANY OBJECT
- CREATE TABLESPACE
- DROP TABLESPACE
- ALTER TABLESPACE
- CREATE LIBRARY
- CREATE ANY DIRECTORY
- GRANT ANY OBJECT PRIVILEGE
- ALTER DATABASE
- ALTER SYSTEM

ANY Clause

Many system privileges contain an ANY clause, which means the privilege applies to all schemas, not just your own. For example, the SELECT ANY TABLE system privilege allows you to retrieve data from all tables and views, including those from schemas owned by other users. The SYS user and users with the DBA role are granted all the ANY privileges; therefore, they can do anything to any data object. You can control the scope of all system privileges, including those with the ANY clause, by using Oracle Database Vault.

ADMIN OPTION

If you grant a system privilege with ADMIN OPTION enabled, you enable the grantee to administer the system privilege and grant it to other users. The SQL syntax for granting system privileges is:

```
SQL> GRANT <system_privilege> TO <grantee_clause> [WITH ADMIN OPTION]
```

System Privileges for Administrators

Privilege	Description
SYSDBA	Perform all administrative tasks in the database, including create and drop a database, open and mount a database, start up and shut down an Oracle database, create an SPFILE, put a database in or remove a database from ARCHIVELOG mode, perform incomplete recovery operations, patch, and migrate. This privilege enables you to connect as the <code>SYS</code> user.
SYSOPER	Perform similar administration tasks as the <code>SYSDBA</code> privilege, but without the ability to look at user data. For example, you can start up and shut down the database, create an SPFILE, and perform complete recovery operations (not incomplete recovery operations).
SYSASM	Start up, shut down, and administer an Automatic Storage Management instance.
SYSBACKUP	Perform backup and recovery operations by using RMAN or SQL*Plus.
SYSDG	Perform Data Guard operations by using the Data Guard Broker or the DGMGRl command-line interface.
SYSKM	Manage Transparent Data Encryption wallet operations.
SYSRAC	Perform day-to-day administration tasks on an Oracle Real Application Clusters (RAC) cluster.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

There are seven special system privileges that are usually granted only to administrators. Anyone who is granted one of these privileges is referred to as a system administrative privileged user (privileged user, for short).

Only users who are granted the `SYSDBA`, `SYSOPER`, `SYSASM`, and `SYSRAC` privileges are allowed to start up and shut down the Oracle database.

The `SYSBACKUP`, `SYSDG`, and `SYSKM` privileges enable you to connect to the database even if the database is not open.

The `SELECT ANY DICTIONARY` system privilege does not permit access to sensitive data dictionary tables, which are owned by the `SYS` schema.

Object Privileges

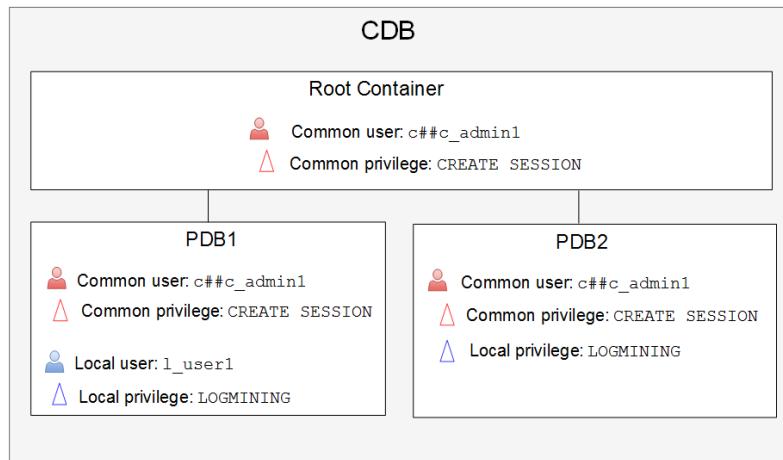
- Object privileges allow a user to perform a particular action on a specific object, such as a table, view, sequence, procedure, function, or package.
- Without specific permission, users can access only their own objects.
- Object privileges can be granted by the owner of an object, by the administrator, or by someone who has been explicitly given permission to grant privileges on the object.
- The SQL syntax for granting object privileges is:

```
GRANT <object_privilege> ON <object> TO <grantee_clause>  
[WITH GRANT OPTION]
```



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Granting Privileges in a Multitenant Environment



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Comparing Common and Local Privileges

A privilege is a right to execute operations that create, modify, or delete structures in an instance or a database or a right to execute operations that manipulate users' objects. Oracle Database has predefined system and object privileges. You do not create them. It is best to grant a privilege only to a user who requires that privilege to accomplish the necessary work. Excessive granting of unnecessary privileges can compromise security.

In a multitenant environment, you can grant a privilege to a user in two ways:

- **Commonly:** You grant the user the privilege in all containers of a CDB. In an application container, a common privilege is granted to a grantee in the application root and application PDBs. To grant a privilege commonly, you must log in to the root container and issue the GRANT command with the CONTAINER=ALL clause. For example:

```

SQL> CONNECT / AS SYSDBA
SQL> GRANT create session TO c##c_admin1 CONTAINER=ALL;
  
```

- **Locally:** You grant the user the privilege in a single PDB only. To grant a privilege locally, log in to the PDB and issue the GRANT command. For example:

```

SQL> CONNECT SYS@PDB1 AS SYSDBA
SQL> GRANT logmining TO l_user1;
  
```

To switch to a different container, a common user must have the SET CONTAINER privilege in the current container. Alternatively, a common user can start a new database session whose initial current container is the container the user wants, relying on the CREATE SESSION privilege in that PDB. Be aware that commonly granted privileges that have been made to common users may interfere with the security configured for individual PDBs.

The diagram in the slide illustrates the following:

- The `c##c_admin1` common user is granted the CREATE SESSION privilege commonly, which applies the privilege to that user in all containers. In PDB2, `c##c_admin1` is also granted the LOGMINING privilege locally. `c##c_admin1` does not have the LOGMINING privilege in any other container.
- The local user `l_user1` exists in PDB1 only and is granted the LOGMINING privilege. If the same user existed in another PDB (as a totally separate user), the LOGMINING privilege would not be applied.

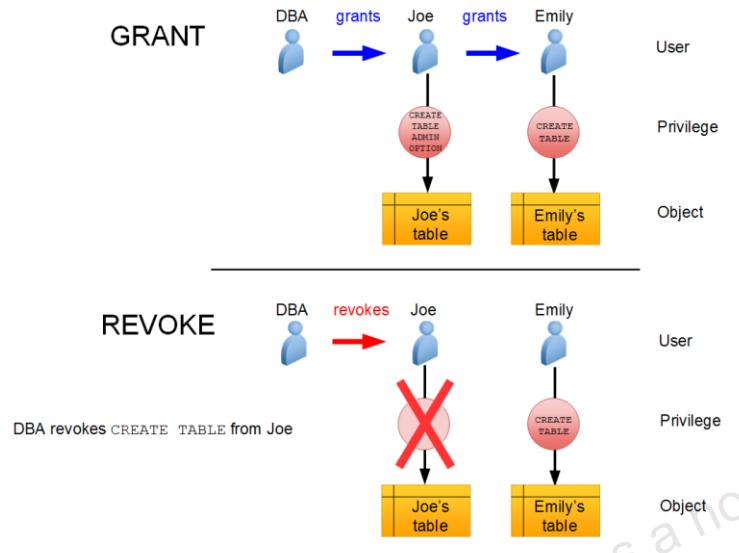
Remember

Step 1: Create a common user when the same user has to perform the same actions in all PDBs in the CDB. Otherwise, create the user as a local user in a PDB.

Step 2: Ask yourself, do you want the common user who exists in each PDB to have the same privilege(s) in the PDBs?

- If yes, then you commonly grant the privilege(s) to the common user. Connect to the CDB as a user who is privileged enough to do it and grant privilege1, privilege2, and so on to the common user by using the CONTAINER=ALL clause.
- If no, then you locally grant the privilege(s) to the common user. Connect to the PDB as a user who is privileged enough to do it and grant privilege1, privilege2, and so on to the common user.

Granting and Revoking System Privileges



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

System privileges that have been granted directly with a `GRANT` command can be revoked by using the `REVOKE` command in SQL*Plus. Users with the `ADMIN OPTION` for a system privilege can revoke the privilege from any other database user. The revoker does not have to be the same user who originally granted the privilege.

There are no cascading effects when a system privilege is revoked, regardless of whether it is given the `ADMIN OPTION`.

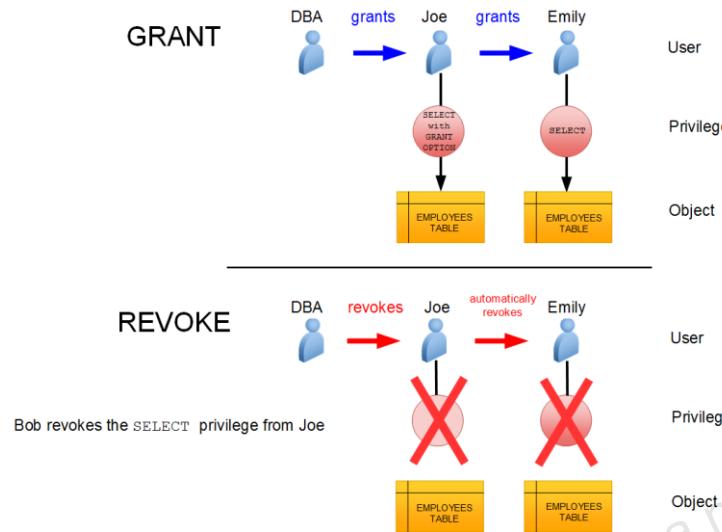
The SQL syntax for revoking system privileges is:

```
SQL> REVOKE <system_privilege> FROM <grantee clause>
```

The diagram in the slide illustrates the following events:

1. The DBA grants the `CREATE TABLE` system privilege to the user **Joe** with `ADMIN OPTION`.
2. Joe creates a table.
3. Joe grants the `CREATE TABLE` system privilege to the user **Emily**.
4. Emily creates a table.
5. The DBA revokes the `CREATE TABLE` system privilege from **Joe**.
6. The result is that Joe's table still exists and he can still access it, but he can't create new tables. Emily's table still exists, and she still has the `CREATE TABLE` system privilege.

Granting and Revoking Object Privileges



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Cascading effects can be observed when revoking a system privilege that is related to a data manipulation language (DML) operation. For example, if the **SELECT ANY TABLE** privilege is granted to a user, and if that user has created procedures that use the table, all procedures that are contained in the user's schema must be recompiled before they can be used again.

Revoking object privileges also cascades when given with **GRANT OPTION**. As a user, you can revoke only those privileges that you have granted. For example, Bob cannot revoke the object privilege that Joe granted to Emily. Only the grantee or a user with the privilege called **GRANT ANY OBJECT PRIVILEGE** can revoke object privileges.

The diagram in the slide illustrates object privileges being revoked. Assume that the following events occur:

1. A DBA grants Joe the **SELECT** object privilege on the **EMPLOYEES** table with the **GRANT OPTION**.
2. Joe grants the **SELECT** privilege on the **EMPLOYEES** table to Emily.
3. The DBA revokes the **SELECT** privilege from Joe.
4. The result is that the revoke takes away Joe's ability to access the **EMPLOYEES** table, and the revoke is cascaded to Emily as well.

Using Roles to Manage Privileges

- Roles:
 - Used to group together privileges and roles
 - Facilitate granting of multiple privileges or roles to users
- Benefits of roles:
 - Easier privilege management
 - Dynamic privilege management
 - Selective availability of privileges



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

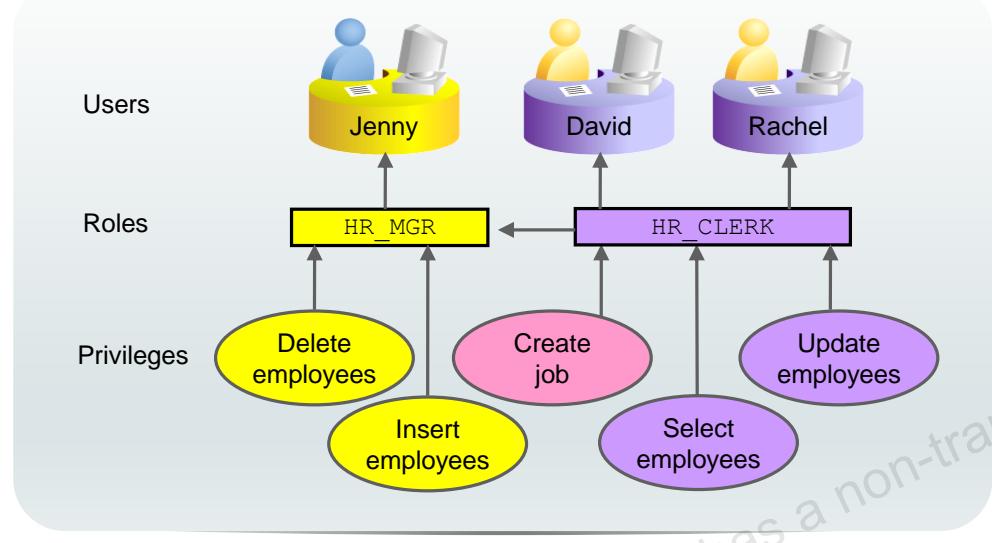
A role is a named group of related privileges that are granted to users or to other roles.

You can use roles to administer database privileges. You can add privileges to a role and grant the role to a user. The user can then enable the role and exercise the privileges granted by the role. A role contains all privileges that are granted to that role and all privileges of other roles that are granted to it.

Roles provide the following benefits with respect to managing privileges:

- **Easier privilege management:** Use roles to simplify privilege management. Rather than granting the same set of privileges to several users, you can grant the privileges to a role and then grant that role to each user.
- **Dynamic privilege management:** If the privileges associated with a role are modified, all users who are granted the role acquire the modified privileges automatically and immediately.
- **Selective availability of privileges:** Roles can be enabled and disabled to turn privileges on and off temporarily. This allows the privileges of the user to be controlled in a given situation.

Assigning Privileges to Roles and Assigning Roles to Users



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In most systems, it is time-consuming and error-prone to grant necessary privileges to each user individually. Oracle software provides for easy and controlled privilege management through roles. Roles are named groups of related privileges that are granted to users or to other roles. Roles are designed to ease the administration of privileges in the database and, therefore, improve security.

Role Characteristics

- Privileges are granted to and revoked from roles as though the role were a user.
- Roles are granted to and revoked from users or other roles as though they were system privileges.
- A role can consist of both system and object privileges.
- A role can be enabled or disabled for each user who is granted the role.
- A role can require a password to be enabled.
- Roles are not owned by anyone, and they are not in any schema.

In the example in the slide, the `SELECT` and `UPDATE` privileges on the `employees` table and the `CREATE JOB` system privilege are granted to the `HR_CLERK` role. `DELETE` and `INSERT` privileges on the `employees` table and the `HR_CLERK` role are granted to the `HR_MGR` role.

The manager is granted the `HR_MGR` role and can now select, delete, insert, and update the `employees` table.

Oracle-Supplied Roles

Account	Description
DBA	Includes most system privileges and several other roles. Do not grant this role to nonadministrators. Users with this role can connect to the CDB or PDB only when it is open.
RESOURCE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
SCHEDULER_ADMIN	CREATE ANY JOB, CREATE EXTERNAL JOB, CREATE JOB, EXECUTE ANY CLASS, EXECUTE ANY PROGRAM, MANAGE SCHEDULER
SELECT_CATALOG_ROLE	SELECT privileges on data dictionary objects

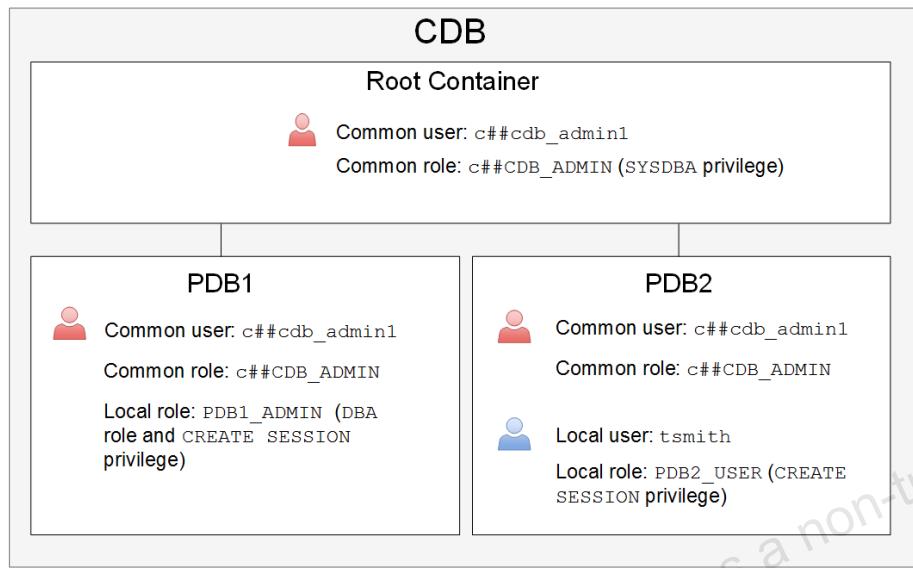


Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The table in the slide lists some commonly used predefined roles in Oracle Database.

You must not alter the privileges granted to Oracle-supplied roles without the assistance of Oracle Support because you may inadvertently disable the needed functionality. The `SYS` and `SYSTEM` accounts have the `DBA` role granted to them by default. In addition, `SYSTEM` is also granted the `AQ_ADMINISTRATOR_ROLE` and `MGMT_USER` roles.

Creating and Granting Roles



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Creating Roles

A role is a group of privileges and/or other roles and is granted to a user or a role. Roles are usually created by administrators; however, there are several Oracle-supplied roles. You can use several tools, including SQL*Plus and Enterprise Manager Express, to create and grant roles.

Roles provide the following benefits with respect to managing privileges:

- **Easier privilege management:** Use roles to simplify privilege management. Rather than granting the same set of privileges to several users, you can grant the privileges to a role and then grant that role to each user.
- **Dynamic privilege management:** If the privileges associated with a role are modified, all users who are granted the role acquire the modified privileges automatically and immediately.
- **Selective availability of privileges:** Roles can be enabled and disabled to turn privileges on and off temporarily. This allows the privileges of the user to be controlled in a given situation.

In a multitenant environment, you can create two types of roles:

- **Common role:** The role is replicated in all current and future containers. All Oracle-supplied predefined roles are common roles. To create a common role, connect to the root container and issue the CREATE ROLE command with the CONTAINER=ALL clause appended to the end. Local users cannot create common roles. For example:

```
SQL> CONNECT / AS SYSDBA
SQL> CREATE ROLE c##c_role1 CONTAINER=ALL;
```

- **Local role:** The role is created in a single PDB and can be used within that PDB only. It does not have any commonly granted privileges. To create a local role, you must connect to the container and issue the CREATE ROLE command. For example:

```
SQL> CONNECT SYS@PDB1 AS SYSDBA  
SQL> CREATE ROLE l_role1;
```

The diagram on the slide illustrates:

- A common role named `c##CDB_ADMIN` is created in all containers. This role consists of the `SYSDBA` privilege and is created for users who need to perform maintenance operations on the entire CDB. The common user `c##c_admin1`, who exists in every container, is granted the `C##CDB_ADMIN` role commonly; meaning, `c##c_admin1` is granted that role in every container.
- A local role named `PDB1_ADMIN` is created and available in `PDB1` only. This role consists of the `DBA` role and the `CREATE SESSION` privilege and is created for users who manage `PDB1`. The common user named `c##c_admin1` is granted this role only in `PDB1`.
- A local role named `PDB2_USER` is created and available in `PDB2` only. The local user named `tsmith`, who exists only in `PDB2`, is granted the `PDB2_USER` role.

Role Characteristics

- Privileges are granted to and revoked from roles as though the role were a user.
- Roles are granted to and revoked from users or other roles as though they were system privileges.
- A role can consist of both system and object privileges.
- A role can be enabled or disabled for each user who is granted the role.
- A role can require a password to be enabled.
- Roles are not owned by anyone, and they are not in any schema.
- The `SYSDBA` privilege is not granted in any role, so you grant it (either commonly or locally) to a user or role.

Assigning Roles

- To assign (grant) a role to a user or another role by using SQL*Plus, use the GRANT command.
- There are two ways to grant a role in a multitenant architecture:
 - Commonly: Grant the role to the user (or role) in all containers.

```
SQL> CONNECT / AS SYSDBA  
SQL> GRANT <common role> TO <common user or role> CONTAINER=ALL;
```

- Locally: Grant the role to a user (or role) in one PDB only.

```
SQL> CONNECT SYS@PDB1 AS SYSDBA  
SQL> GRANT <common or local role> TO <common or local user>;
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Comparing Granting Roles Commonly and Locally in a Multitenant Architecture

There are two ways to grant a role in a multitenant architecture:

- **Commonly:** You grant the role to the user (or role) in all containers. Do this if the user needs to perform the same operation in all containers. To grant a role commonly, log in to the root container and issue the GRANT command to a common user (or common role) with the CONTAINER=ALL clause. The role that you grant must be a common role before you can grant it commonly. For example:

```
SQL> CONNECT / AS SYSDBA  
SQL> GRANT <common role> TO <common user or role> CONTAINER=ALL;
```

- **Locally:** You grant the role to a user (or role) in one PDB only. To grant a role locally, log in to the PDB where the user exists and issue the GRANT command without the CONTAINER=ALL clause. The role that you grant can be a common or local role. For example:

```
SQL> CONNECT SYS@PDB1 AS SYSDBA  
SQL> GRANT <common or local role> TO <common or local user>;
```

Making Roles More Secure

- Roles are usually enabled by default, which means that if a role is granted to a user, then that user can exercise the privileges given to the role immediately.
- Default roles are assigned to the user at connect time.
- Use the following security measures to make roles more secure:
 - Make a role nondefault.
 - Use role authentication.
 - Create application roles.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can use the following security measures to make roles more secure:

- Make a role nondefault. Not all privileges and roles granted to a user need to be available to that user at logon. You can configure specific roles to be enabled by default and set the other roles to be nondefault (disabled). The user can enable the nondefault roles when needed by using the `SET ROLE` command. For example, a user `tsmith` is granted both the `HRCLERK` and `HRMANAGER` roles. The `HRMANAGER` role has more privileges and contains the `HRCLERK` role. You configure `tsmith`'s default role to be `HRCLERK`, so when `tsmith` logs on to the PDB, she automatically is granted the `HRCLERK` role. When she needs to operate as the manager, she enables the `HRMANAGER` role.
 - To configure default roles for a user, use the `ALTER USER` command with the `DEFAULT ROLE` clause. In the `DEFAULT ROLE` clause, you cannot specify a role that is a member of another role (that is, a subrole).
 - Note that the `SET ROLE` command disables roles you don't specify in the command.
- Use role authentication. Have a role require additional authentication by using the `IDENTIFIED` clause to indicate that a user must be authorized by a specified method before the role is enabled with the `SET ROLE` statement. The default authentication for a role is `None`. You can define role authentication in Enterprise Manager Cloud Control, but not in Enterprise Manager Database Express.
- Create application roles. Create secure application roles that a user must enable by executing a PL/SQL procedure successfully. The PL/SQL procedure can check things, such as the user's network address, the program that the user is running, the time of day, and other elements needed, to properly secure a group of permissions.

```
CREATE ROLE secure_application_role IDENTIFIED USING  
<security_procedure_name>
```

Note: Oracle Database Vault also includes secure application roles, which are different from the secure application roles described in this course. See *Oracle Database Vault Administrator's Guide* for details on secure application roles in Oracle Database Vault.

Revoking Roles and Privileges

You can use the REVOKE statement to:

- Revoke system privileges from users and roles
- Revoke roles from users, roles, and program units
- Revoke object privileges for a particular object from users and roles



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Profiles and Users

- Users are assigned only one profile at a time.
- Profiles:
 - Control resource consumption
 - Manage account status and password expiration
- `RESOURCE_LIMIT` must be set to `TRUE` before profiles can impose resource limitations.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Profiles impose a named set of resource limits on database usage and instance resources. Profiles also manage the account status and place limitations on users' passwords (length, expiration time, and so on). Every user is assigned a profile and may belong to only one profile at any given time. If users have already logged in when you change their profile, the change does not take effect until their next login.

The `DEFAULT` profile serves as the basis for all other profiles. Limitations for a profile can be implicitly specified (as in CPU/Session), can be unlimited (as in CPU/Call), or can reference whatever setting is in the `DEFAULT` profile (as in Connect Time).

Profiles cannot impose resource limitations on users unless the `RESOURCE_LIMIT` initialization parameter is set to `TRUE`. With `RESOURCE_LIMIT` at its default value of `FALSE`, profile resource limitations are ignored. Profile password settings are always enforced.

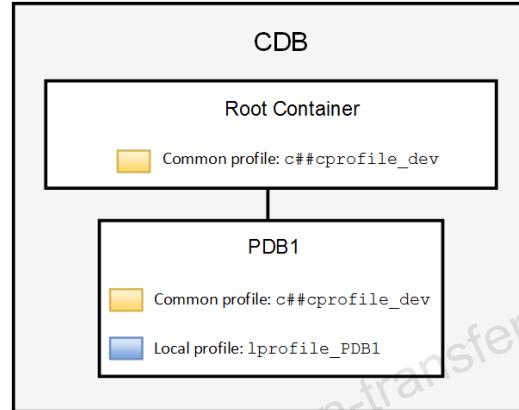
Creating Profiles in a Multitenant Architecture

- Common profile:

```
SQL> CREATE PROFILE c##cprofile_dev
  2  limit ... CONTAINER=ALL;
```

- Local profile:

```
SQL> CREATE PROFILE lprofile_PDB1
  2  limit ... ;
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Creating Profiles

Oracle recommends that you use profiles to create and manage password security (you can use the `CREATE PROFILE` command) and use Database Resource Manager to manage machine resources because it offers a more flexible means of managing and tracking resource use.

In a multitenant environment, you can create two types of profiles:

- **Common profile:** The profile is replicated in all current and future containers. To create a common profile by using SQL*Plus, log in to the root container and issue the `CREATE PROFILE` command with the `CONTAINER=ALL` clause. For example:

```
SQL> CONNECT / AS SYSDBA
SQL> CREATE PROFILE c##cprofile_dev limit ... CONTAINER=ALL;
```

- **Local profile:** The profile is created in a single PDB and can be used within that PDB only. To create a local profile by using SQL*Plus, log in to the PDB and issue the `CREATE PROFILE` command without the `CONTAINER=ALL` clause. For example:

```
SQL> CONNECT SYS@PDB1 AS SYSDBA
SQL> CREATE PROFILE lprofile_PDB1 limit ... ;
```

In the diagram in the slide, the common profile named `c##cprofile_dev` is created commonly at the CDB level. The `CREATE` operation is replicated in all containers, including the root container where it was initially created. Consequently, the same profile `c##cprofile_dev` is created in PDB1. The profile named `lprofile_PDB1` is created locally in PDB1 and exists only in PDB1.

Dropping or Changing Profiles

In Enterprise Manager Database Express, you cannot drop a profile that is used by users. However, if you drop a profile with the CASCADE option (for example, in SQL*Plus), all users who have that profile are automatically assigned the DEFAULT profile.

If users have already logged in when you change their profile, the change does not take effect until their next login.

Profile Parameters: Resources

- In a profile, you can control:
 - CPU resources: May be limited to a per-session or per-call basis
 - Network and memory resources (Connect time, Idle time, Concurrent sessions, Private SGA)
- Disk I/O resources: Limit the amount of data a user can read at the per-session level or per-call level.
- Profiles cannot impose resource limitations on users unless the `RESOURCE_LIMIT` initialization parameter is set to TRUE. With `RESOURCE_LIMIT` at its default value of FALSE, profile resource limitations are ignored.
- Profiles also allow composite limits, which are based on weighted combinations of CPU/session, reads/session, connect time, and private SGA.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

CPU Resources

CPU resources may be limited on a per-session or per-call basis. A CPU/Session limitation of 1,000 means that if any individual session that uses this profile consumes more than 10 seconds of CPU time (CPU time limitations are in hundredths of a second), that session receives the following error message and is logged off:

`ORA-02392: exceeded session limit on CPU usage, you are being logged off.`

A per-call limitation does the same thing, but instead of limiting the user's overall session, it prevents any single command from consuming too much CPU. If CPU/Call is limited and the user exceeds the limitation, the command aborts. The user receives an error message, such as:

`ORA-02393: exceeded call limit on CPU usage`

Network and Memory Resources

Each database session consumes system memory resources and (if the session is from a user who is not local to the server) network resources. You can specify the following:

- **Connect Time:** Indicates for how many minutes a user can be connected before being automatically logged off.
- **Idle Time:** Indicates for how many minutes a user's session can remain idle before being automatically logged off. Idle time is calculated for the server process only. It does not take into account application activity. The `IDLE_TIME` limit is not affected by long-running queries and other operations.
- **Concurrent Sessions:** Indicates how many concurrent sessions can be created by using a database user account.

- **Private SGA:** Limits the amount of space consumed in the System Global Area (SGA) for sorting, merging bitmaps, and so on. This restriction takes effect only if the session uses a shared server configuration.

Disk I/O Resources

Disk I/O resources limit the amount of data a user can read at the per-session level or per-call level. Reads/Session and Reads/Call place a limitation on the total number of reads from both memory and the disk. This can be done to ensure that no I/O-intensive statements overuse memory and disks.

Profile Parameters: Locking and Passwords

- In a profile, specific parameters control account locking, password aging and expiration, and password history.
- Profile password settings are always enforced.
- Account locking enables automatic locking of accounts for a set duration when users fail to log in to the system in the specified number of attempts or when accounts sit inactive for a predefined number of days (users have not attempted to log in to their accounts).
- Password aging and expiration enables user passwords to have a lifetime, after which the passwords expire and must be changed.
- Password history checks the new password to ensure that the password is not reused for a specified amount of time or a specified number of password changes.
- Password complexity verification makes a complexity check on the password to verify that it meets certain rules.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Account Locking

Account locking enables automatic locking of accounts for a set duration when users fail to log in to the system in the specified number of attempts or when accounts sit inactive for a predefined number of days (users have not attempted to log in to their accounts). Configure the following profile parameters:

- `FAILED_LOGIN_ATTEMPTS` specifies the number of failed login attempts before the lockout of the account.
- `PASSWORD_LOCK_TIME` specifies the number of days for which the account is locked after the specified number of failed login attempts.
- `INACTIVE_ACCOUNT_TIME` specifies the number of days an account can be inactive before it is locked.

Password Aging and Expiration

Password aging and expiration enables user passwords to have a lifetime, after which the passwords expire and must be changed. Configure the following profile parameters:

- `PASSWORD_LIFE_TIME` determines the lifetime of the password in days, after which the password expires.
- `PASSWORD_GRACE_TIME` specifies a grace period in days for changing the password after the first successful login after the password has expired.

Expiring passwords and locking the `SYS`, `SYSMAN`, and `DBSNMP` accounts prevent Enterprise Manager from functioning properly. The applications must catch the “password expired” warning message and handle the password change; otherwise, the grace period expires and the user is locked out without knowing the reason.

Password History

Password history checks the new password to ensure that the password is not reused for a specified amount of time or a specified number of password changes. Configure one of the following parameters:

- `PASSWORD_REUSE_TIME` specifies that a user cannot reuse a password for a given number of days.
- `PASSWORD_REUSE_MAX` specifies the number of password changes that are required before the current password can be reused.
- `PASSWORD_VERIFY_FUNCTION` checks for password complexity for the `SYS` user.

Recall that the values of the profile parameters are either set or inherited from the `DEFAULT` profile.

If both password history parameters have a value of `UNLIMITED`, Oracle Database ignores both. The user can reuse any password at any time, which is not a good security practice. If both parameters are set, password reuse is allowed, but only after meeting both conditions. The user must have changed the password the specified number of times, and the specified number of days must have passed since the old password was last used. For example, the profile of user `ALFRED` has `PASSWORD_REUSE_MAX` set to 10 and `PASSWORD_REUSE_TIME` set to 30. User `ALFRED` cannot reuse a password until he has reset the password 10 times and until 30 days have passed since the password was last used. If one parameter is set to a number and the other parameter is specified as `UNLIMITED`, then the user can never reuse a password.

Password Complexity Verification

Password complexity verification makes a complexity check on the password to verify that it meets certain rules. The check must ensure that the password is complex enough to provide protection against intruders who may try to break into the system by guessing the password.

The `PASSWORD_VERIFY_FUNCTION` parameter names a PL/SQL function that performs a password complexity check before a password is assigned. Password verification functions must be owned by the `SYS` user and return a Boolean value (`TRUE` or `FALSE`). A model password verification function is provided in the `utlpwdmg.sql` script found in the following directories:

- UNIX and Linux platforms: `$ORACLE_HOME/rdbms/admin`
- Windows platforms: `%ORACLE_HOME%\rdbms\admin`

You can create complexity functions and run the `utlpwdmg.sql` script to create the `VERIFY_FUNCTION_11g`, `ORA12C_VERIFY_FUNCTION`, and `ORA12C_STRONG_VERIFY_FUNCTION` prebuilt complexity functions.

Oracle-Supplied Password Verification Functions

- Complexity verification checks that each password is complex enough to provide reasonable protection against intruders who try to break into the system by guessing passwords.
- You can create your own password verification functions.
- Oracle Database provides the following functions that you can create by executing the `utlpwdmg.sql` script:
 - `ORA12c_VERIFY_FUNCTION`
 - `ORA12c_STRONG_VERIFY_FUNCTION`
 - `VERIFY_FUNCTION_11g`
- These functions must be owned by the `SYS` user.
- Password complexity checking is not enforced for the `SYS` user.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Complexity Verification

Complexity verification checks that each password is complex enough to provide reasonable protection against intruders who try to break into the system by guessing passwords. Using a complexity verification function forces users to create strong, secure passwords for database user accounts. You must ensure that the passwords for your users are complex enough to provide reasonable protection against intruders who try to break into the system by guessing passwords.

Oracle-Supplied Functions

You can create your own password verification functions; however, Oracle Database provides the following functions that you can create by executing the `utlpwdmg.sql` script, located in `$ORACLE_HOME/rdbms/admin`:

- `ORA12c_VERIFY_FUNCTION`: This function performs the minimum complexity checks such as checking for a minimum password length and that the password is not the same as the username.
- `ORA12c_STRONG_VERIFY_FUNCTION`: This function provides a stronger password complexity function that takes into consideration recommendations from the US Department of Defense Database Security Technical Implementation Guide.
- `VERIFY_FUNCTION_11g`: This function performs minimum complexity checks such as checking for a minimum password length and that the password is not the same as the username. This function was provided with Oracle Database 11g.

In addition to creating the functions, the `utlpwdmg.sql` script also changes the `DEFAULT` profile with the following `ALTER PROFILE` command:

```
ALTER PROFILE default LIMIT  
PASSWORD_LIFE_TIME 180  
PASSWORD_GRACE_TIME 7  
PASSWORD_REUSE_TIME UNLIMITED  
PASSWORD_REUSE_MAX UNLIMITED  
FAILED_LOGIN_ATTEMPTS 10  
PASSWORD_LOCK_TIME 1  
PASSWORD_VERIFY_FUNCTION ora12c_verify_function;
```

Assigning Profiles

There are two ways to assign a profile:

- **Commonly:** The profile assignment is replicated in all current and future containers.

```
SQL> CONNECT / AS SYSDBA
```

```
SQL> ALTER USER <common user> PROFILE <common profile>
CONTAINER=ALL;
```

- **Locally:** The profile assignment occurs in one PDB (stand-alone or application container) only.

```
SQL> CONNECT SYS@PDB1 AS SYSDBA
```

```
SQL> ALTER USER <common or local user> PROFILE <common or local
profile>;
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

There are two ways to assign a profile to a user:

- **Commonly:** The profile assignment is replicated in all current and future containers. The user must be a common user and the profile must be a common profile. Do this if the user needs the same profile in all containers. To assign a profile commonly, log in to the root container and issue the `ALTER USER` command with the `PROFILE` and `CONTAINER=ALL` clauses.
- **Locally:** The profile assignment occurs in one PDB (stand-alone or application container) only. The user can be common or local, and the profile can be common or local. To assign a profile locally, log in to the PDB where the user exists and issue the `ALTER USER` command with the `PROFILE` clause. Exclude the `CONTAINER=ALL` clause.

Assigning Quotas

- A quota is a space allowance in a given tablespace.
- By default, a user has no quota on any of the tablespaces.
- Database accounts that need quota are those that own database objects (for example, accounts for applications).
- Only those activities that use space in a tablespace count against quota.
 - Oracle server checks quota when you create or extend a segment.
 - Activities that don't use space don't impact quota (example: `CREATE VIEW`).
 - You can be granted permission to use objects without needing any quota.
- Quota is not needed for assigned temporary tablespaces or undo tablespaces.
- A user's quota is replenished when he drops objects (with the `PURGE` clause) or purges his objects in the recycle bin.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Assigning Quotas

You have three options for providing quota for a user on a tablespace:

- UNLIMITED
- Value
- UNLIMITED TABLESPACE system privilege



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You have three options for providing a quota for a user on a tablespace:

- UNLIMITED: Allows the user to use as much space as is available in the tablespace
- Value: Number of kilobytes or megabytes that the user can use. This does not guarantee that the space is set aside for the user. This value can be larger or smaller than the current space that is available in the tablespace.
- UNLIMITED TABLESPACE system privilege: Overrides all individual tablespace quotas and gives the user unlimited quota on all tablespaces, including SYSTEM and SYSAUX. This privilege must be granted with caution.

You must not provide a quota to users on the SYSTEM or SYSAUX tablespaces. Typically, only the SYS and SYSTEM users are able to create objects in the SYSTEM or SYSAUX tablespaces.

Applying the Principle of Least Privilege

- The principle of least privilege means that a user must be given only those privileges that are required to efficiently complete a task.
- This reduces the chances of users modifying or viewing data (either accidentally or maliciously) that they do not have the privilege to modify or view.
- Ways to apply the principle of least privilege:
 - Protect the data dictionary
 - Revoke unnecessary privileges from PUBLIC
 - Use access control lists (ACLs) to control network access
 - Restrict access to OS directories
 - Limit users with administrative privileges
 - Restrict remote database authentication
 - Enable unified auditing



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The following are ways that you can apply the principle of least privilege.

- **Protect the data dictionary:** The `O7_DICTIONARY_ACCESSIBILITY` parameter is set by default to `FALSE`. You must not allow this to be changed without a very good reason because it prevents users with the `ANY TABLE` system privileges from accessing the data dictionary base tables. It also ensures that the `SYS` user can log in only as `SYSDBA`.
- **Revoke unnecessary privileges from PUBLIC:** Several packages are extremely useful to applications that need them, but require proper configuration to be used securely. `PUBLIC` is granted execute privilege on the following packages: `UTL_SMTP`, `UTL_TCP`, `UTL_HTTP`, and `UTL_FILE`.
- **Use access control lists (ACLs) to control network access:** In Oracle Database, network access is controlled by an ACL that may be configured to allow certain users access to specific network services. Network access is denied by default. An ACL must be created to allow network access. File access through `UTL_FILE` is controlled at two levels:
 - At the OS level with permissions on files and directories
 - In the database by `DIRECTORY` objects that allow access to specific file system directories. The `DIRECTORY` object may be granted to a user for read or for read and write. Execute privileges on other PL/SQL packages should be carefully controlled.

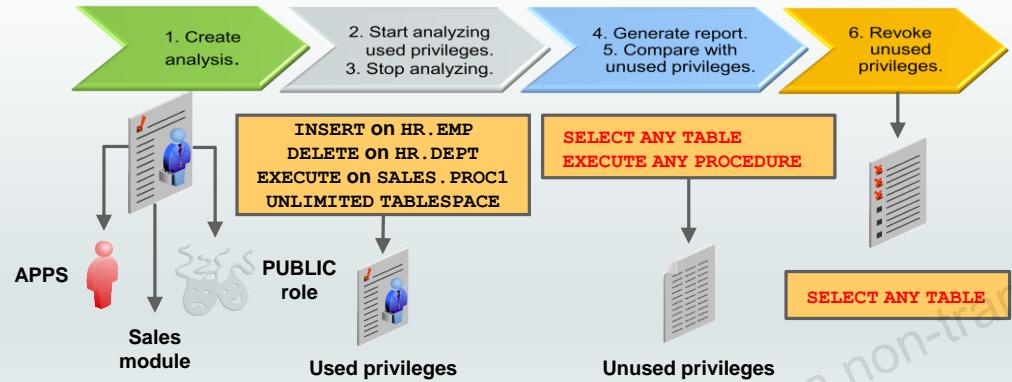
The more powerful packages that may potentially be misused include:

- `UTL_SMTP`: Permits arbitrary email messages to be sent by using the database as a Simple Mail Transfer Protocol (SMTP) mail server. Use the ACL to control which machines may be accessed by which users.

- UTL_TCP: Permits outgoing network connections to be established by the database server to any receiving or waiting network service. Thus, arbitrary data can be sent between the database server and any waiting network service. Use the ACL to control access.
 - UTL_HTTP: Allows the database server to request and retrieve data via HTTP. Granting this package to a user may permit data to be sent via HTML forms to a malicious website. Limit access by using the ACL.
 - UTL_FILE: If configured improperly, allows text-level access to any file on the host operating system. When properly configured, this package limits user access to specific directory locations.
- **Restrict access to OS directories:** The DIRECTORY object inside the database enables DBAs to map directories to OS paths and to grant privileges on those directories to individual users.
 - **Limit users with administrative privileges:** Do not provide database users more privileges than necessary. Nonadministrators must not be granted the DBA role. To implement least privilege, restrict the following types of privileges:
 - Grants of system and object privileges
 - SYS-privileged connections to the database, such as SYSDBA and SYSOPER
 - Other DBA-type privileges, such as DROP ANY TABLE
 - **Restrict remote database authentication:** The REMOTE_OS_AUTHENT parameter is set to FALSE by default. It must not be changed unless all clients can be trusted to authenticate users appropriately. In the remote authentication process:
 - The database user is authenticated externally
 - The remote system authenticates the user
 - The user logs in to the database without further authentication
- Note:** Always test your applications thoroughly if you have revoked privileges.
After applying the principle of least privilege, you can track changes that users make in the database.
- **Unified auditing:**
 - Auditing is the monitoring and recording of configured database actions from both database users.
 - Unified Auditing centralizes all audit records in one place.
 - You can configure auditing for both successful and failed activities and include or exclude specific users from the audit.
 - In a multitenant environment, you can audit individual actions of PDBs or individual actions in the entire CDB.
 - Auditing is enabled by default for the SYS user (instance parameter AUDIT_SYS_OPERATIONS = TRUE).
 - All audit records are written to the unified audit trail in a uniform format and are made available through the UNIFIED_AUDIT_TRAIL view.

Privilege Analysis

- Analyze used privileges to revoke unnecessary privileges.
- Use the DBMS_PRIVILEGE_CAPTURE package.



ORACLE

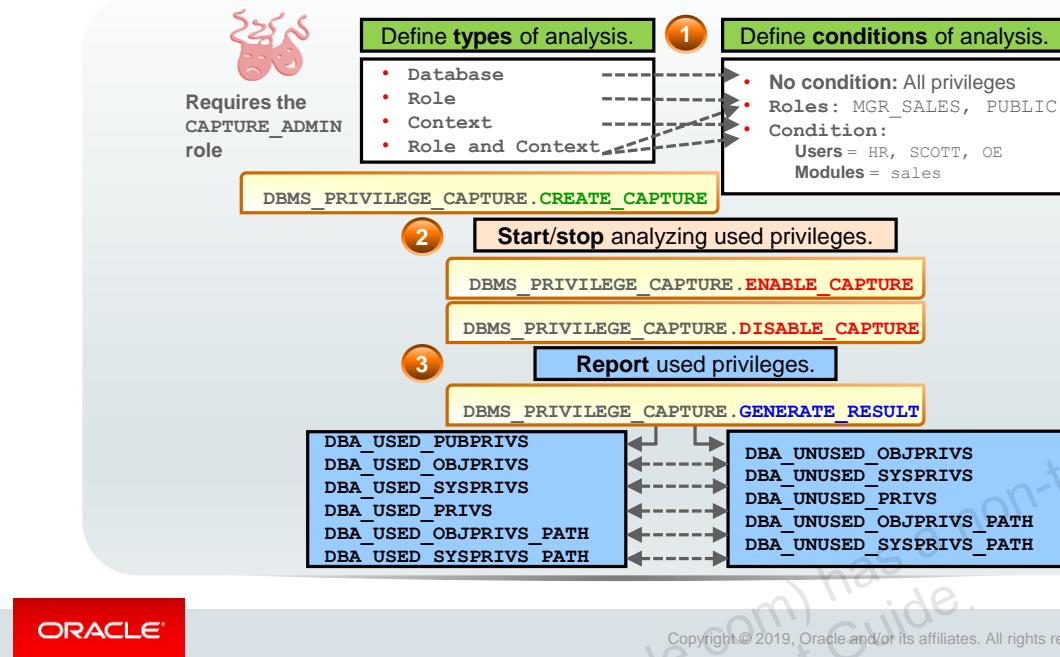
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

A major concern in many databases is that existing database and application users have excessive privileges. Excessive privileges violate the principle of least privilege. To achieve the least privilege principle, unused privileges need to be identified.

The DBMS_PRIVILEGE_CAPTURE package can be used to analyze used privileges.

You can use a privilege analysis policy to identify object and system privileges used to run an application module or to execute certain SQL statements or privileges used by defined roles. You can generate reports of used and unused privileges during the analysis period. The report helps the security officer revoke unnecessary privileges by comparing the used and unused granted privilege lists.

Privilege Analysis Flow



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When creating an analysis, you first define the targeted objects to be analyzed in used privileges. You do that by setting the type of analysis:

- **Database analysis:** If no condition is given, it analyzes the used privileges (except privileges used by administrative users) within the whole database.
- **Role analysis:** If roles are defined, it analyzes the privileges exercised through any given role. For example, if you create a privilege analysis policy to analyze on PUBLIC, the privileges that are directly and indirectly granted to PUBLIC are analyzed when they are used.
- **Context-specific analysis:** If the contexts are defined, it analyzes the privileges that are used through a given application module or specified contexts.

Different conditions can be combined with “AND” and/or “OR” Boolean operators.

Because the created policy is not enabled by default, your next step is to enable the policy to start analyzing used privileges. After a certain time, you stop analyzing.

Your third step is to generate a report. Reporting includes two types of results:

- Used privileges visible in DBA_USED_xxx and DBA_USED_xxx_PATH views
- Unused privileges visible in DBA_UNUSED_xxx and DBA_UNUSED_xxx_PATH views

Summary

In this lesson, you should have learned how to:

- Create database users
- Grant privileges to database users
- Create and grant roles to users or other roles
- Revoke privileges and roles from users and other roles
- Create and assign profiles to users
- Explain the various authentication options for users
- Assign quota to users
- Apply the principle of least privilege



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 9: Overview

- 9-1: Creating Common and Local Users
- 9-2: Creating a Local User for an Application
- 9-3: Granting a Local Role (DBA) to PDBADMIN
- 9-4: Using EM Express to Create a Local Profile
- 9-5: Using EM Express to Create Local Roles
- 9-6: Using EM Express to Create Local Users
- 9-7: Configuring a Default Role for a User
- 9-8: Exploring OS and Password File Authentication



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Creating PDBs

The ORACLE logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the methods and tools used to create PDBs
- Create PDBs from seed by using SQL*Plus
- Clone PDBs by using SQL*Plus
- Unplug and plug in PDBs by using SQL*Plus
- Drop PDBs by using SQL*Plus



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When working with a multitenant Oracle database, you have the opportunity to spread things out. This means that each application can have its own PDB. You don't have to put all your data from all applications into one PDB. So, part of your planning process as a database administrator is to determine how many PDBs you think you'll need. This lesson will show you some fundamental ways to create PDBs and how to move them around between CDBs by unplugging and plugging them.

Methods and Tools to Create PDBs

- Methods to create PDBs:
 - Create a PDB by using the seed
 - Create a PDB from a non-CDB
 - Clone an existing PDB or non-CDB
 - Plug an unplugged PDB into a different CDB
 - Relocate a PDB to a different CDB
 - Create a PDB as a proxy PDB
- Tools to create PDBs:
 - SQL*Plus
 - SQL Developer
 - Enterprise Manager Cloud Control
 - DBCA—Create a PDB from seed or by using the unplug/plug method.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

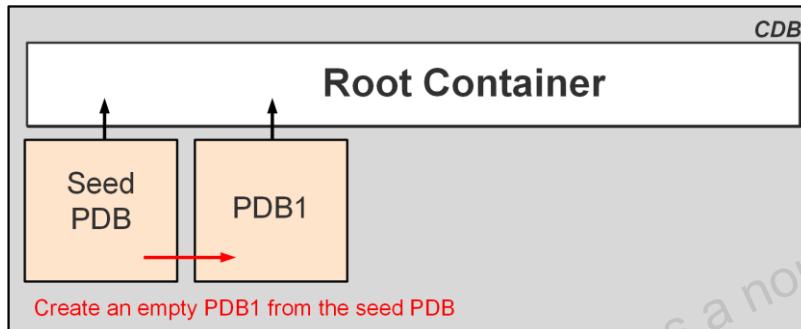
You can use the following methods to create a PDB:

- **Create a PDB by using the seed:** Create a PDB in a CDB by using the files of the CDB seed or application seed. This technique copies the files associated with the seed to a new location and associates the copied files with the new PDB.
- **Create a PDB from a non-CDB:** Create a PDB by adopting a non-CDB into a PDB. You can use the DBMS_PDB package to create an unplugged PDB from an Oracle Database 12c non-CDB. You can then plug the unplugged PDB into the CDB.
- **Clone an existing PDB or non-CDB:** Create a PDB by cloning a source PDB or non-CDB. A source can be a PDB in the local CDB, a PDB in a remote CDB, a PDB in a local or remote application container, or a non-CDB. This technique copies the files associated with the source to a new location and associates the copied files with the new PDB.
- **Plug an unplugged PDB into a CDB:** Create a PDB by using the XML metadata file that describes the PDB and the files associated with the PDB to plug it into the CDB.
- **Relocate a PDB to a different CDB:** Create a PDB by relocating it from one CDB to another. This technique moves the files associated with the PDB to a new location.
- **Create a PDB as a proxy PDB:** Create a PDB as a proxy PDB by referencing a different PDB with a database link. The referenced PDB can be in the same CDB as the proxy PDB, or it can be in a different CDB.

Note: This course focuses on how to use SQL*Plus to create a PDB from seed and clone a PDB.

Creating PDBs from Seed

- You can create a new empty PDB by using the seed PDB as a template.
- Every CDB has a seed PDB.
- To create a PDB from seed with SQL*Plus, use the `CREATE PLUGGABLE DATABASE` statement.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

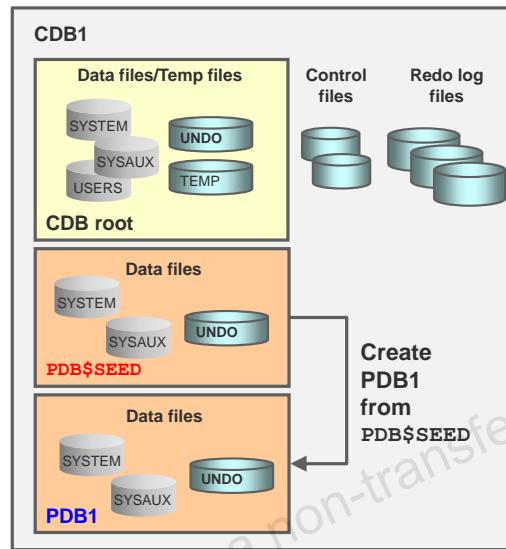
To create a PDB from seed with SQL*Plus, you use the `CREATE PLUGGABLE DATABASE` statement. The creation of a new PDB from the seed is nearly instantaneous. This action copies the data files from the `READ-ONLY` seed PDB to the target directory defined in the `CREATE PLUGGABLE DATABASE` statement. It creates tablespaces such as `SYSTEM` to store a full catalog, including metadata pointing to Oracle-supplied objects, and `SYSAUX` for local auxiliary data. It creates default schemas and common users that exist in seed PDB, `SYS` who continues to have all super user privileges, and `SYSTEM` who can administer the PDB. A new default service is created for the PDB.

Prerequisites for using the `CREATE PLUGGABLE DATABASE` statement include:

- You must be connected to a CDB and the current container must be the root.
- You must have the `CREATE PLUGGABLE DATABASE` system privilege.
- The CDB in which the PDB is being created must be in `READ WRITE` mode.

Creating a New PDB from PDB\$SEED

- Copies the data files from PDB\$SEED data files
- Creates the SYSTEM, SYSAUX, and UNDO tablespaces
- Creates a full catalog including metadata pointing to Oracle-supplied objects
- Creates common users:
 - SYS
 - SYSTEM
- Creates a local user (PDBA), granted local PDB_DBA role
- Creates a new default service



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The creation of a new PDB from the CDB seed is nearly instantaneous. The operation copies the data files from the READ-ONLY seed PDB to the target directory defined in the CREATE PLUGGABLE DATABASE statement.

It creates tablespaces such as SYSTEM to store a full catalog, including metadata pointing to Oracle-supplied objects, SYSAUX for local auxiliary data, and UNDO for local undo segments.

It creates default schemas and common users that exist in the CDB seed, SYS who continues to have all superuser privileges, and SYSTEM who can administer the PDB.

It creates a local user (the PDBA), who is granted a local PDB_DBA role. Until the PDB SYS user grants privileges to the local PDB_DBA role, the new PDBA cannot perform any other operation than connecting to the PDB.

A new default service is also created for the PDB.

Examples: Creating a PDB from Seed

- Create a new PDB from the seed by using FILE_NAME_CONVERT:

```
SQL> CREATE PLUGGABLE DATABASE pdb1
  ADMIN USER admin1 IDENTIFIED BY p1 ROLES=(CONNECT)
  FILE_NAME_CONVERT = ('PDB$SEEDdir', 'PDB1dir');
```

- Set the PDB_FILE_NAME_CONVERT initialization parameter and then create the PDB:

```
SQL> CREATE PLUGGABLE DATABASE pdb1
  ADMIN USER pdb1_admin IDENTIFIED BY p1 ROLES=(CONNECT);
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If you do not use Oracle Managed Files (OMF), connect to the CDB root as a common user with the CREATE PLUGGABLE DATABASE system privilege and execute the CREATE PLUGGABLE DATABASE statement as shown in the slide. The ADMIN USER clause defines the PDBA user created in the new PDB with the CONNECT and PDB_DBA roles (empty role). The FILE_NAME_CONVERT clause first designates the source directory of the CDB seed data files and then the destination directory for the new PDB data files.

You can also set the PDB_FILE_NAME_CONVERT initialization parameter to both the source directory of the CDB seed data files and the target directory for the new PDB data files.

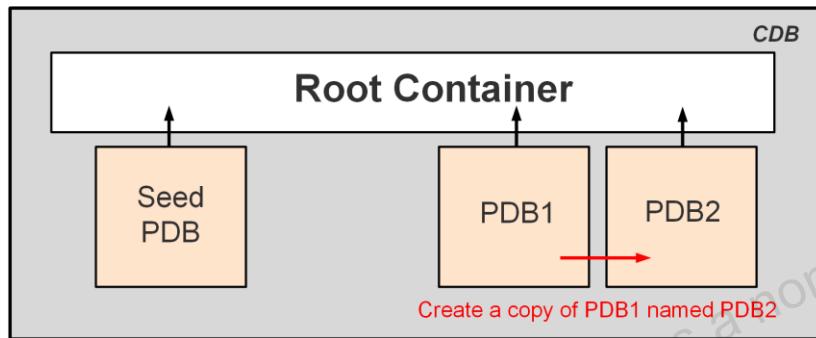
If you are using OMF, set the DB_CREATE_FILE_DEST initialization parameter to a target directory for the data files of the new PDB.

When the statement completes, use views to verify that the PDB is correctly created. The CDB_PDBS view displays the list of the PDBs, and the CDB_TABLESPACES view displays the list of the tablespaces of the new PDB (SYSTEM, SYSAUX, UNDO).

The CDB_PDBS view shows the STATUS of the new PDB: it is NEW. The PDB has never been opened. It must be opened in READ WRITE or RESTRICTED mode for Oracle to perform the processing needed to complete the integration of the PDB into the CDB and mark it NORMAL. An error will be returned if an attempt is made to open the PDB read-only.

Cloning PDBs

- Cloning is copying a source PDB from a CDB and plugging the copy into the same CDB or another CDB.
- Example: PDB1 is cloned as PDB2 in the same CDB. The seed PDB, while present in the CDB, is not used.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Cloning is suitable for the following situations:

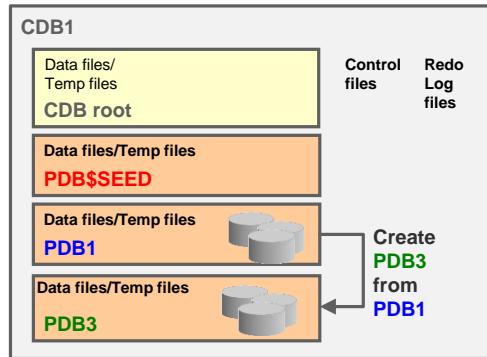
- You want to test the application patch of your production PDB. You first clone your production application in a cloned PDB, patch the cloned PDB, and test it.
- You want to diagnose performance issues or perform performance regression tests on your application. Because you cannot perform this operation in parallel with the production in the same database, you clone the PDB into another CDB.

To clone a PDB with SQL*Plus, you use the `CREATE PLUGGABLE DATABASE` statement.

Prerequisites for using the `CREATE PLUGGABLE DATABASE` statement to clone a PDB include:

- You must be connected to a CDB and the current container must be the root.
- You must have the `CREATE PLUGGABLE DATABASE` system privilege.
- The CDB in which the PDB is being created must be in `READ/WRITE` mode.
- You must put the PDB being cloned into `READ-ONLY` mode before you can clone it.

Cloning Regular PDBs



PDB3 owns:

- SYSTEM, SYSAUX, UNDO tablespaces
- Full catalog
- SYS, SYSTEM common users
- Same local administrator name
- New service name

1. Define how Oracle will find the location of the data files:

- In init.ora, set `DB_CREATE_FILE_DEST= 'PDB3dir'`
- In init.ora, set `PDB_FILE_NAME_CONVERT='PDB1dir', 'PDB3dir'`
- Using the `CREATE_FILE_DEST= 'PDB3dir'` clause

2. Connect to the CDB root to close `PDB1`.

3. Clone `PDB3` from `PDB1`.

```
SQL> CREATE PLUGGABLE DATABASE pdb3 FROM pdb1
      CREATE FILE DEST = 'PDB3dir';
```

4. Open `PDB3` in read/write mode.

```
SQL> ALTER PLUGGABLE DATABASE pdb3 OPEN;
```

Note: Cloning metadata only with NO DATA

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

This technique copies a source PDB from a CDB and plugs the copy into a CDB. The source PDB is in the local CDB.

The steps to clone a PDB within the same CDB are:

1. Define the location for the data files of the new PDB:

- In init.ora, set `DB_CREATE_FILE_DEST= 'PDB3dir'` (OMF) or `PDB_FILE_NAME_CONVERT= 'PDB1dir', 'PDB3dir'` (non-OMF).
- Use the `CREATE_FILE_DEST` clause during the `CREATE PLUGGABLE DATABASE` statement (OMF).
- Use the `FILE_NAME_CONVERT= ('pdb1dir', ' pdb3dir')` clause to define the directory of the source files to copy from PDB1 and the target directory for the new files of PDB3 (non-OMF).

2. Connect to the CDB root as a common user with the `CREATE PLUGGABLE DATABASE` privilege.

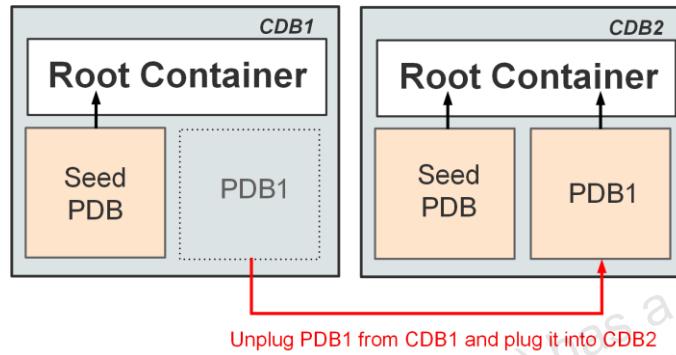
3. Use the `CREATE PLUGGABLE DATABASE` command to clone the PDB `PDB3` from `PDB1`.

4. Then open the new `PDB3` with the command `ALTER PLUGGABLE DATABASE OPEN`.

Note: NO DATA allows PDB metadata cloning, which is an option that can be used for cloning an empty PDB and later importing data for testing.

Unplugging and Plugging in PDBs

- Unplugging a PDB is disassociating the PDB from its CDB.
- Plugging in a PDB is associating a PDB with a CDB.
- You can plug a PDB into the same or another CDB.
- Example: PDB1 is unplugged from CDB1 and plugged into CDB2.



ORACLE®

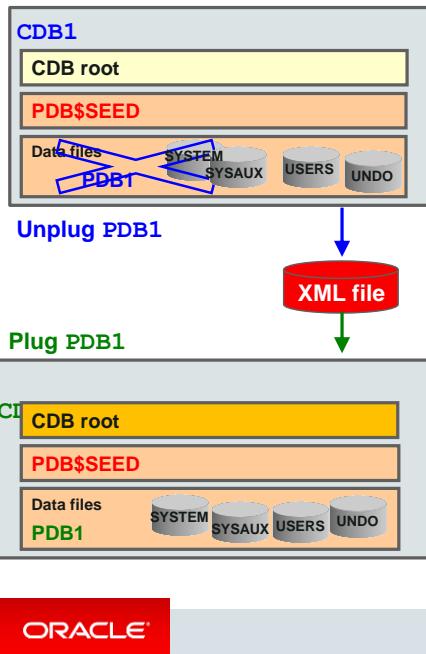
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Unplugging and plugging is suitable for the following situations:

- You have to upgrade a PDB to the latest Oracle version, but you do not want to apply it on all PDBs. Instead of upgrading a CDB from one release to another, you can unplug a PDB from one Oracle Database release and then plug it into a newly created CDB from a later release.
- You want to test the performance of the CDB without a particular PDB. You unplug the PDB, test the performance without the PDB, and, if necessary, replug the PDB into the CDB.
- You want to maintain a collection of PDB “gold images” as unplugged PDBs.

You use the `ALTER PLUGGABLE DATABASE` statement to unplug a PDB from a CDB. To unplug a PDB, you must first close it and then generate an XML manifest file. The XML file contains the names and full paths of the tablespaces and data files of the unplugged PDB. That information is then used by the plugging operation. You use the `CREATE PLUGGABLE DATABASE` statement to plug a PDB into a CDB.

Plugging an Unplugged Regular PDB into a CDB



Unplug **PDB1** from **CDB1**:

1. Connect to **CDB1** as a common user.
2. Verify that **PDB1** is closed.

```
SQL> ALTER PLUGGABLE DATABASE pdb1
      UNPLUG INTO 'xmlfile1';
```

3. Drop **PDB1** from **CDB1**.

Plug **PDB1** into **CDB2**:

1. Connect to **CDB2** as a common user.
2. Use the DBMS_PDB package to check the compatibility of **PDB1** with **CDB2**.

```
SQL> CREATE PLUGGABLE DATABASE pdb1
      USING 'xmlfile1' NOCOPY;
```

3. Open **PDB1** in read/write mode.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can create a PDB in a CDB by using the unplugging/plugging method.

Unplugging a PDB disassociates the PDB from a CDB. You unplug a PDB when you want to move the PDB to a different CDB or when you no longer want the PDB to be available.

The first step is to unplug PDB1 from CDB1. The second step is to plug PDB1 into CDB2.

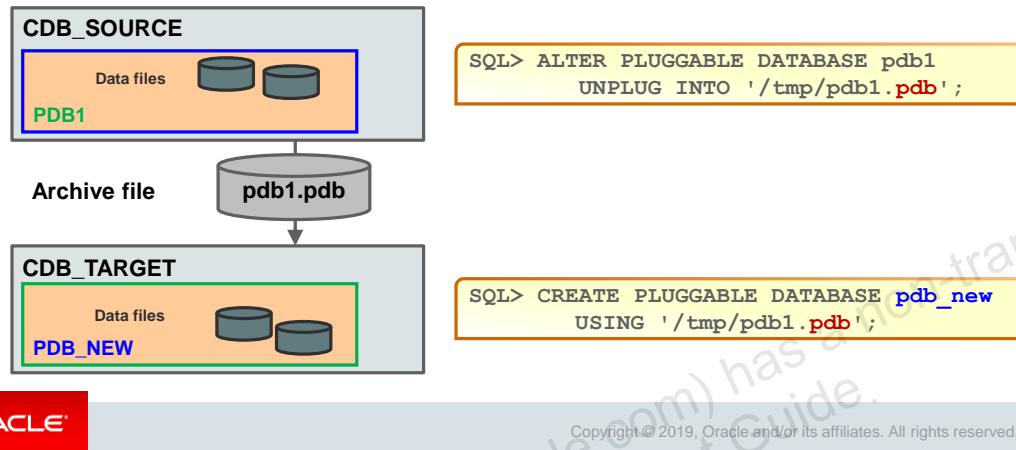
To unplug PDB1 from CDB1, first connect to the source CDB root and check that the PDB is closed by using the V\$PDBS view. Then use ALTER PLUGGABLE DATABASE with the UNPLUG clause to specify the database to unplug and the XML file to unplug it into. The STATUS in CDB_PDBS of the unplugged PDB is UNPLUGGED. A PDB must be dropped from the CDB before it can be plugged back into the same CDB. If the PDB is plugged into another CDB, the PDB does not need to be dropped if the data files are copied.

Before plugging PDB1 into CDB2, you can optionally check whether the unplugged PDB is compatible with CDB2 with the DBMS_PDB.CHECK_PLUG_COMPATIBILITY function.

To plug PDB1 into CDB2, connect to CDB2 root and use CREATE PLUGGABLE DATABASE pdb1 USING 'xmlfile1.xml'. The last step is to open the PDB.

Plugging Using an Archive File

- Unplugging a PDB into a single archive file includes:
 - XML file
 - Data files
- Plugging the PDB requires only the archive file.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When a PDB is unplugged, all the data files associated with the PDB along with the PDB manifest must be copied or moved individually over to the remote server where it will be plugged into another CDB. You can choose to create a single PDB archive file, a compressed file with the .pdb extension, which contains the PDB manifest and all the data files when unplugging a PDB. When plugging in a PDB, the presence of a .pdb file is interpreted, and the PDB is plugged into the CDB. You can choose to run the PDB plug-in compatibility test directly on the PDB archive without extracting the PDB manifest file from the archive.

This feature provides ease of managing the unplugging and plugging of PDBs across CDBs.

Dropping PDBs

- When you drop a PDB, you remove all references to it and its data files in the control file of the CDB.
- Archived logs and backups associated with the dropped PDB are not deleted in case you later want to recover the PDB.
- You can also use Oracle Recovery Manager (RMAN) to delete archived logs and backups.
- You use the `DROP PLUGGABLE DATABASE` statement to drop a pluggable database (PDB).

– Example:

```
SQL> DROP PLUGGABLE DATABASE SALES_PDB INCLUDING DATAFILES;
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You use the `DROP PLUGGABLE DATABASE` statement to drop a pluggable database (PDB).

- Specify `KEEP DATAFILES` to retain the data files associated with the PDB after the PDB is dropped. The temp file for the PDB is deleted because it is no longer needed. This is the default. Keeping data files may be useful in scenarios where a PDB that is unplugged from one CDB is plugged into another CDB, with both CDBs sharing storage devices.
- Specify `INCLUDING DATAFILES` to delete the data files associated with the PDB being dropped. The temp file for the PDB is also deleted.

Prerequisites for using the `DROP PLUGGABLE DATABASE` statement to drop PDBs include:

- You must be connected to a CDB.
- The current container must be the root. You must be authenticated as `SYSDBA` or as `SYSOPER`, and the `SYSDBA` or `SYSOPER` privilege must be either granted to you commonly or granted to you locally in the root and locally in the PDB you want to drop.
- To specify `KEEP DATAFILES` (the default), the PDB you want to drop must be unplugged.
- To specify `INCLUDING DATAFILES`, the PDB you want to drop must be in mounted mode or unplugged.

Summary

In this lesson, you should have learned how to:

- Describe the methods and tools used to create PDBs
- Create PDBs from seed by using SQL*Plus
- Clone PDBs by using SQL*Plus
- Unplug and plug in PDBs by using SQL*Plus
- Drop PDBs by using SQL*Plus



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 10: Overview

- 10-1: Creating a PDB from Seed
- 10-2: Cloning a PDB
- 10-3: Unplugging and Plugging in a PDB
- 10-4: Dropping a PDB



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Creating Master Encryption Keys for PDBs

The ORACLE logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the implementation of master encryption keys for PDBs
- Create and activate a master encryption key for a new PDB



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Encryption in Database Cloud Service



- Database Cloud Service databases include a key management framework that stores and manages keys and credentials used to encrypt data in the database data files and in backups.
- The key management framework includes:
 - The keystore (referred to as a wallet in Oracle Database 11g and previous releases) to securely store Transparent Data Encryption (TDE) master encryption keys
 - The management framework to securely and efficiently manage the keystore and key operations for various database components

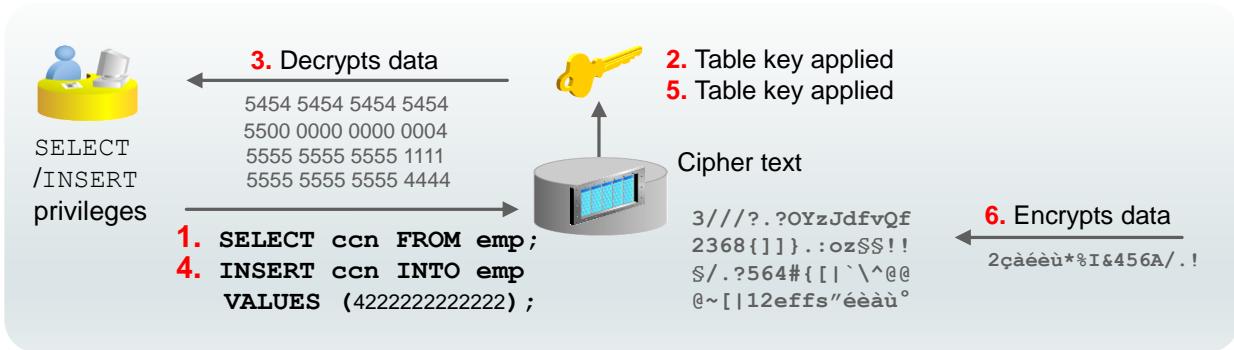
ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Database Cloud Service databases include a key management framework that stores and manages keys and credentials used to encrypt data in the database data files and in backups.

The key management framework includes the keystore (referred to as a wallet in Oracle Database 11g and previous releases) to securely store Transparent Data Encryption (TDE) master encryption keys and the management framework to securely and efficiently manage keystore and key operations for various database components. TDE is the underlying mechanism used for default tablespace encryption and encrypted backups in Database Cloud Service.

Transparent Data Encryption (TDE): Overview



- Encrypts data in data files, redo log files, archived redo log files, and backup files
- Encrypts data in memory (only for column encryption)
- Manages keys automatically
- Does not require application changes

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Transparent Data Encryption (TDE) is available with Oracle Advanced Security and provides easy-to-use protection for your data without requiring changes to your applications.

In Database Cloud Service, High Performance Service and Extreme Performance Service include Oracle Advanced Security, enabling you to use all the features provided with it. Standard Edition Service and Enterprise Edition Service do not include Oracle Advanced Security. However, all editions include the tablespace encryption feature by default.

TDE allows you to encrypt sensitive data in individual columns or entire tablespaces without having to manage encryption keys. TDE does not affect access controls, which are configured by using database roles, secure application roles, system and object privileges, views, Virtual Private Database (VPD), Oracle Database Vault, and Oracle Label Security. Any application or user that previously had access to a table will still have access to an identical encrypted table.

TDE is designed to protect data in storage, but does not replace proper access control.

TDE is transparent to existing applications. Encryption and decryption occur at different levels depending on whether it is tablespace or column level, but in either case, encrypted values are not displayed and are not handled by the application. For example, with TDE, applications designed to display a 16-digit credit card number do not have to be recoded to handle an encrypted string that may have many more characters.

TDE eliminates the ability of anyone who has direct access to the data files to gain access to the data by circumventing the database access control mechanisms. Even users with access to the data file at the operating system level cannot see the data unencrypted. TDE stores the master key outside the database in an external security module, thereby minimizing the possibility of both personally identifiable information (PII) and encryption keys being compromised. TDE decrypts the data only after database access mechanisms have been satisfied.

Components of TDE

- Key architecture
 - Two-tier architecture: A unified master encryption key stored in an external security module is used to encrypt the table key or tablespace key.
 - Low overhead re-key operation: Some security regulations require periodical changes of encryption keys.
- External security module
 - Software keystore
 - Hardware keystore (HSM)
- Algorithm support



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

TDE applies the principle of defense in depth in its design. The key architecture is a two-tier system. The master key is stored in an external security module. This is either an Oracle software keystore or a hardware keystore also called hardware security module (HSM). In releases before Oracle Database 12c, the software keystore was called a "wallet." This external store is protected by a password, operating system permissions, and encryption. The master encryption key is used to encrypt the table key (for column encryption) and the tablespace key (for tablespace encryption). The table key or tablespace key is then used to encrypt the data.

In Oracle Database Cloud Service, the external security module is an Oracle software keystore.

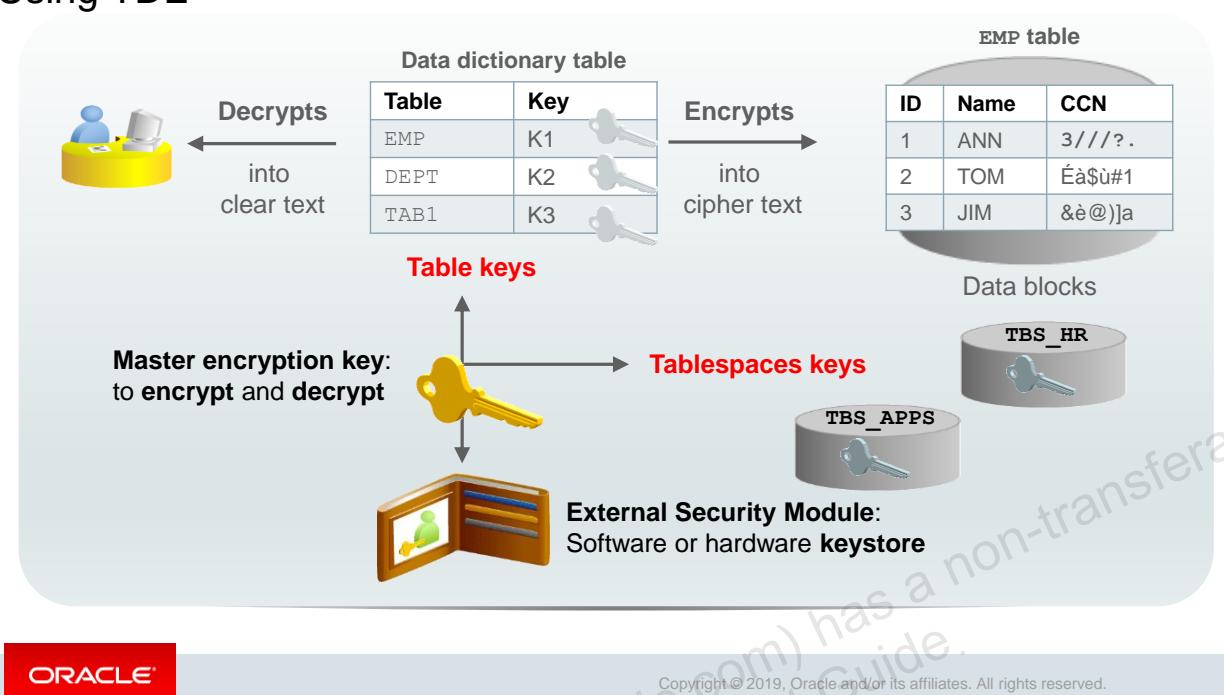
Some security regulations require a periodic change of encryption keys. This change of keys means that the items that are encrypted must be decrypted with the old key and encrypted with the new key. This is also called re-keying.

A major advantage of the two-tier architecture is that the table-level keys can be re-keyed by changing the master key. This automatically causes table-level keys to be re-encrypted by using the new master key, but the table-level keys remain unchanged. So the data does not require re-keying. This operation meets the Payment Card Industry requirement for re-keying, with a minimum overhead.

With TDE, you can specify different encryption algorithms to be used at the table or tablespace level. The available algorithms are 3DES168, AES128, AES192, and AES256. The default is AES192 for column encryption and AES128 for tablespace encryption.

AES128 is the default encryption algorithm for Oracle Database Cloud Service.

Using TDE



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

TDE enables encryption for sensitive data in columns without requiring users or applications to manage the encryption key. This freedom can be extremely important when addressing, for example, regulatory compliance issues. There is no need to use views to decrypt data because the data is transparently decrypted when a user has passed the necessary access control checks. Security administrators have the assurance that the data on disk is encrypted, yet handling encrypted data is transparent to applications.

The external security module is implemented through an API that allows a variety of possible key storage solutions. The default external security module is a software keystore. Hardware security modules (HSMs) from several vendors are also supported for storage of the master keys. TDE support of HSM varies by database version and whether column encryption or tablespace encryption is being used.

In Oracle Database Cloud Service, the external security module is an Oracle software keystore.

Defining the Keystore Location

The location of the keystore file is specified in an entry in the \$ORACLE_HOME/network/admin/sqlnet.ora file.



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

TDE creates a key for each table that uses encrypted columns and each encrypted tablespace. The table key is stored in the data dictionary, and the tablespace keys are stored in the tablespace data files. Both tablespace and table keys are encrypted with a master key. There is one master key for the database. The master key is stored in a PKCS#12 software keystore or a PKCS#11-based HSM, outside the database. For the database to use TDE, a keystore must exist.

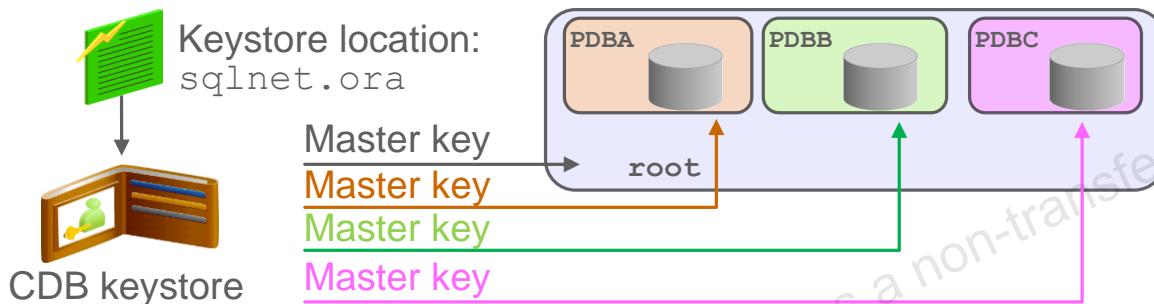
The location of the keystore file used to store the encryption master key is specified in an entry in the \$ORACLE_HOME/network/admin/sqlnet.ora file:

```
ENCRYPTION_WALLET_LOCATION =  
(SOURCE =  
  (METHOD = FILE)  
  (METHOD_DATA =  
    (DIRECTORY = /u01/app/oracle/admin/ORCL/tde_wallet)))
```

In Oracle Database Cloud Service, the location of the keystore file is set during service instance creation.

CDB and PDB Master Encryption Keys

- There is one master encryption key per PDB to encrypt PDB data.
- The master key must be transported from the source database keystore to the target database keystore when a PDB is moved from one host to another.
- After creating or plugging in a new PDB, you must create and activate a master encryption key for the PDB.



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In a multitenant container database (CDB), the root container and each pluggable database (PDB) has its own master key used to encrypt data in the PDB. The master encryption keys are stored in a single keystore used by all containers.

The master key must be transported from the source database keystore to the target database keystore when a PDB is moved from one host to another.

Oracle Database Cloud Service: After creating or plugging in a new PDB, you must create and activate a master encryption key for the PDB.

Do You Need to Create and Activate a Master Encryption Key?

1. Set the container to the PDB.

```
SQL> ALTER SESSION SET CONTAINER = pdb;
```

2. Query the STATUS column in V\$ENCRYPTION_WALLET.

```
SQL> SELECT wr1_parameter, status, wallet_type  
2  FROM v$encryption_wallet;
```

3. If STATUS contains a value of OPEN_NO_MASTER_KEY, create and activate the master encryption key (shown in the next slide).



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

To determine whether you need to create and activate an encryption key for a PDB, set the container to the PDB and query the STATUS column in V\$ENCRYPTION_WALLET. If the STATUS column contains a value of OPEN_NO_MASTER_KEY, you need to create and activate a master encryption key for the PDB.

Creating and Activating a Master Encryption Key

1. Close the auto-login keystore in the root container and then reopen it as a password keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE close;
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE open
  2 IDENTIFIED BY keystore-password CONTAINER = all;
```

2. Set the container to the PDB. Create and activate a master encryption key in the PDB.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY USING TAG 'tag'
  2 IDENTIFIED BY keystore-password
  3 WITH BACKUP USING 'backup_identifier';
```

3. Query V\$ENCRYPTION_WALLET again, verifying that STATUS is OPEN.

```
SQL> SELECT wrl_parameter, status, wallet_type
  2 FROM v$encryption_wallet;
```

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Before creating the new master encryption key for the PDB, you must first close the auto-login keystore in the root container and then reopen it as a password keystore.

Then set the container to the PDB for which you need to create the master encryption key.

Create and activate a master encryption key in the PDB. You can use the optional USING TAG clause to associate a tag with the new master encryption key. Specify the WITH BACKUP clause, and optionally the USING '*backup_identifier*' clause, to create a backup of the keystore before the new master encryption key is created.

Summary

In this lesson, you should have learned how to:

- Describe the implementation of master encryption keys for PDBs
- Create and activate a master encryption key for a new PDB



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 11: Overview

- 11-1: Creating and Activating an Encryption Key
- 11-2: Creating and Activating the Encryption Key for PDB2



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Creating and Managing Tablespaces

The ORACLE logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Explain how table data is stored in the database
- Use SQL*Plus to:
 - Create and drop tablespaces
 - Alter tablespaces
 - View tablespace information
- Implement Oracle Managed Files (OMF)
- Use SQL*Plus to move and rename online data files
- Implement tablespace encryption

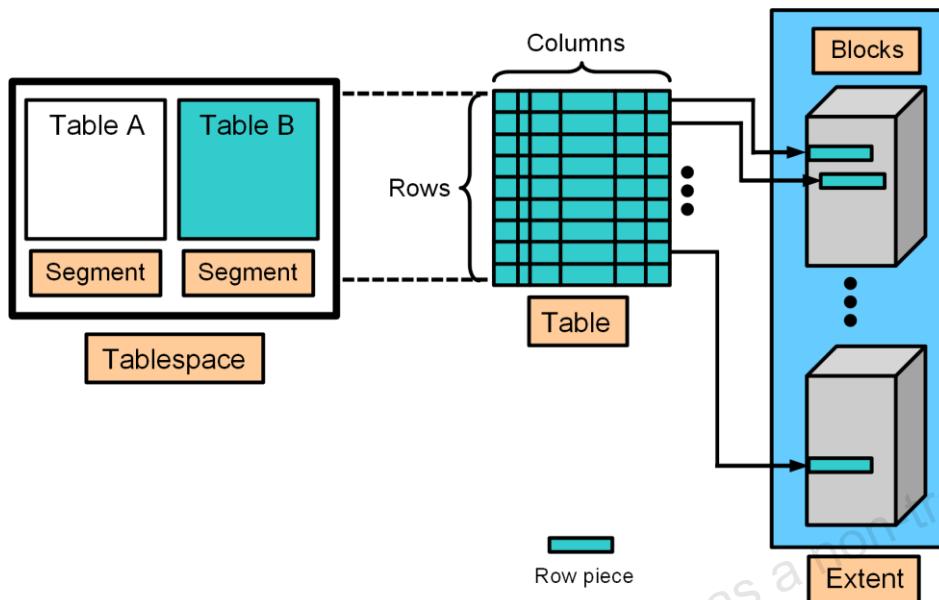


ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The size of a PDB can be described as the sum of all its tablespaces. As you manage your PDBs over time, you may need to enlarge them as usage increases. You can do that by creating new tablespaces, adding data files to existing smallfile tablespaces, increasing the size of data files, and providing for the dynamic growth of your data files. All these activities can be performed with SQL statements, although if you prefer working with a GUI, you can use Enterprise Manager Cloud Control or Enterprise Manager Database Express (EM Express).

How Table Data Is Stored



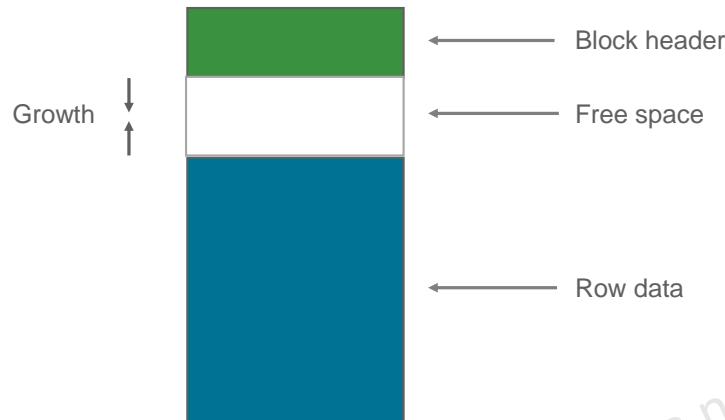
ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When a table is created, a logical "segment" is created to hold its data, as illustrated in the slide. A tablespace contains a collection of segments.

Logically, a table contains rows of column values. A row is ultimately stored in a database block in the form of a row piece (also illustrated in the slide). It is called a row piece because, under some circumstances, the entire row may not be stored in one place. This happens when an inserted row is too large to fit into a single block (chained row) or when an update causes an existing row to outgrow the available free space of the current block (migrated row). Row pieces are also used when a table has more than 255 columns. In this case, the pieces may be in the same block (intra-block chaining) or across multiple blocks.

Database Block Content



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

A database block contains a block header, row data, and free space, as illustrated in the slide.

- A block header contains the segment type (such as table or index), data block address, table directory, row directory, and transaction slots of approximately 23 bytes each, which are used when modifications are made to rows in the block. The block header grows downward from the top.
- Row data is the actual data for the rows in the block. Row data space grows upward from the bottom.
- Free space is in the middle of the block, enabling the header and the row data space to grow when necessary. Row data takes up free space as new rows are inserted or as columns of existing rows are updated with larger values. Examples of events that cause header growth:
 - Row directories that need more row entries
 - More transaction slots required than initially configured

Initially, the free space in a block is contiguous. However, deletions and updates may fragment the free space in the block. The free space in the block is coalesced by the Oracle server when necessary.

Creating Tablespaces

- A tablespace is an allocation of space in the database that can contain schema objects.
- Create a tablespace with the `CREATE TABLESPACE` statement or a graphical interface, such as EM Express.
- You can create three types of tablespaces:
 - Permanent tablespace: Contains persistent schema objects. Objects in permanent tablespaces are stored in data files.
 - Undo tablespace: Is a type of permanent tablespace used by Oracle Database to manage undo data if you are running your database in automatic undo management mode. Oracle strongly recommends that you use automatic undo management mode rather than using rollback segments for undo.
 - Temporary tablespace: Contains schema objects only for the duration of a session. Objects in temporary tablespaces are stored in temp files.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Prerequisites for CREATE TABLESPACE

Before you can create a tablespace, you must create a database to contain it, and the database must be open. You must also have the `CREATE TABLESPACE` system privilege. To create the `SYSAUX` tablespace, you must have the `SYSDBA` system privilege.

File Name and Size

You must include the `DATAFILE` or `TEMPFILE` clause when you create a tablespace. Use this clause to specify the name and location of the data file or temp file. A tablespace must have at least one data file or temp file. You must also specify an initial file size.

You can include the `AUTOEXTEND ON` clause to automatically extend the file when it is full. In this case, you'll need to specify an increment amount and a maximum file size, which can be unlimited. Remember, the size of the file is limited by the physical media on which it resides.

You can include the `BIGFILE` or `SMALLFILE` clause to override the default tablespace type (permanent or temporary) for the database. If you omit this clause, then Oracle Database uses the current default tablespace type.

- A bigfile tablespace contains only one data file (or temp file), which can contain up to approximately 4 billion blocks. Bigfile tablespaces are used with extremely large databases, in which Automatic Storage Management (ASM) or other logical volume managers support the striping or redundant array of independent disks (RAID) and dynamically extensible logical volumes. For bigfile tablespaces, you can specify only one data file in the `DATAFILE` clause or one temp file in the `TEMPFILE` clause.
- A smallfile tablespace is a traditional Oracle tablespace, which can contain 1022 data files or temp files, each of which can contain up to approximately 4 million blocks.

Availability

You can include the `ONLINE` or `OFFLINE` clause to make the tablespace available (or not available) immediately after creation to users who have been granted access to the tablespace. `ONLINE` is the default. The data dictionary view `DBA_TABLESPACES` indicates whether each tablespace is online or offline. This clause cannot be used with temporary tablespaces.

Block Size

You can include the `BLOCKSIZE` clause to specify a nonstandard block size. To specify this clause, the `DB_CACHE_SIZE` and at least one `DB_nK_CACHE_SIZE` parameter must be set, and the integer you specify in this clause must correspond with the setting of one `DB_nK_CACHE_SIZE` parameter setting. You cannot specify nonstandard block sizes for a temporary tablespace or if you intend to assign this tablespace as the temporary tablespace for any users. If you don't specify a block size, the database will use the default 8 KB block size for the tablespace.

Extent Management

You can include the `EXTENT MANAGEMENT` clause to specify how the extents of the tablespace will be managed.

- `AUTOALLOCATE` specifies that the tablespace is system managed. Users cannot specify an extent size. You cannot specify `AUTOALLOCATE` for a temporary tablespace.
- `UNIFORM` value specifies that the tablespace is managed with uniform extents of `SIZE` bytes. The default `SIZE` is 1MB. All extents of temporary tablespaces are of uniform size, so this keyword is optional for a temporary tablespace. However, you must specify `UNIFORM` to specify `SIZE`. You cannot specify `UNIFORM` for an undo tablespace.

If you don't specify `AUTOALLOCATE` or `UNIFORM`, then the default is `UNIFORM` for temporary tablespaces and `AUTOALLOCATE` for all other types of tablespaces. If you do not specify the `EXTENT MANAGEMENT` clause, then Oracle Database interprets the `MINIMUM EXTENT` clause and the `DEFAULT STORAGE` clause to determine extent management.

Also, with the `EXTENT MANAGEMENT` clause, you can specify where the metadata for allocated and unallocated extents is to be stored, either in the data dictionary (`DICTONARY`) or in the tablespace itself (`LOCAL`). Tablespaces that record extent allocation in the dictionary are called dictionary-managed tablespaces. Tablespaces that record extent allocation in the tablespace header are called locally-managed tablespaces.

Logging

You can include the `LOGGING` clause to specify the default logging attributes of all tables, indexes, materialized views, materialized view logs, and partitions within a tablespace. The logging attribute controls whether certain DML operations are logged in the redo log file (`LOGGING`) or not (`NOLOGGING`). The default is `LOGGING`. This clause is not valid for a temporary or undo tablespace.

If logging is not enabled, any direct loads using SQL*Loader and direct load `INSERT` operations are not written to the redo log, and the objects are thus unrecoverable in the event of data loss. When an object is created without logging enabled, you must back up those objects if you want them to be recoverable. Choosing not to enable logging can have a significant impact on the ability to recover objects in the future. Use with caution.

You can use the `FORCE LOGGING` clause to put the tablespace into `FORCE LOGGING` mode. Oracle Database will log all changes to all objects in the tablespace except changes to temporary segments, overriding any `NOLOGGING` setting for individual objects. The database must be open and in `READ WRITE` mode.

Segment Space Management

You can include the `SEGMENT MANAGEMENT` clause to specify whether Oracle Database should track the used and free space in the segments in the tablespace using by free lists or bitmaps (`AUTO`) or not (`MANUAL`).

- **AUTO:** The Oracle Database server will use bitmaps to manage the free space in segments. The bitmap describes the status of each data block in a segment with respect to the amount of space in the block that is available for inserting rows. As more or less space becomes available in a data block, the new state is reflected in the bitmap. With bitmaps, the Oracle Database manages free space more automatically. As a result, this form of space management is called Automatic Segment Space Management (ASSM).
- **MANUAL:** You want to use free lists for managing free space in segments. Free lists are lists of data blocks that have space available for inserting rows. This form of managing space in segments is called manual segment space management because of the need to specify and tune the `PCTUSED`, `FREELISTS`, and `FREELIST GROUPS` storage parameters for schema objects created in the tablespace. This is supported for backward compatibility; it is recommended that you use ASSM.

The `SEGMENT MANAGEMENT` clause applies to permanent, locally managed tablespaces only and is not valid for temporary tablespaces.

Data Segment Compression

When you create a tablespace, you can specify that all tables and indexes, or their partitions, created in a tablespace are compressed by default.

Data segment compression is disabled by default. Enabling data segment compression can save disk space usage, reduce memory use in the buffer cache, and speed up query execution during reads. There is, however, a cost in CPU overhead for data loading and DML. It is especially useful in online analytical processing (OLAP) systems, where there are lengthy read-only operations, but can also be used in online transaction processing (OLTP) systems.

Creating Permanent Tablespaces in a CDB

- Tablespace creation during CDB creation:
 - With DBCA: USERS tablespace created in the CDB root
 - With CREATE DATABASE statement with USER_DATA TABLESPACE clause: Your defined tablespace created in the CDB root
- Create a permanent tablespace in the CDB root:

```
SQL> CONNECT system@cdb1
SQL> CREATE TABLESPACE tbs_CDB_users
    DATAFILE '/u1/app/oracle/oradata/cdb/cdb_users01.dbf' SIZE 100M;
```

- Create a permanent tablespace in a PDB:

```
SQL> CONNECT system@PDB1
SQL> CREATE TABLESPACE tbs_PDB1_users
    DATAFILE '/u1/app/oracle/oradata/cdb/pdb1/users01.dbf' SIZE 100M;
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In a non-CDB, all the tablespaces belong to one database.

In a CDB, one set of tablespaces belong to the CDB root, and each PDB has its own set of tablespaces.

The CREATE TABLESPACE command should be familiar. The change in its behavior in a CDB is that the tablespace is created in the container where the command is executed.

Separating the data files into different directories by PDB can help determine which files belong to which PDB, though it is not necessary.

You can use Oracle ASM storage to manage your disk storage.

The USER_DATA TABLESPACE clause in the CREATE DATABASE command allows you to specify a default tablespace other than USERS when using DBCA to create a database. This tablespace will also be used for XDB options.

Altering and Dropping Tablespaces

- When you create a tablespace, it is initially a read/write tablespace.
- Use the `ALTER TABLESPACE` statement to take a tablespace offline or online, add data files or temp files to it, or make it a read-only tablespace.
- A tablespace can be in one of three different statuses or states:
 - Read Write
 - Read Only
 - Offline with one of the following options:
 - NORMAL
 - TEMPORARY
 - IMMEDIATE
- Add space to an existing tablespace by either adding data files to the tablespace or changing the size of an existing data file.
- Use the `DROP TABLESPACE` statement to drop a tablespace and its contents from the database if you no longer need its content.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Changing the Status

A tablespace can be in one of three different statuses or states. Any of the following three states may not be available because their availability depends on the type of tablespace:

- Read Write:** The tablespace is online and can be read from and written to.
- Read Only:** Specify read-only to place the tablespace in transition read-only mode. In this state, existing transactions can be completed (committed or rolled back), but no further data manipulation language (DML) operations are allowed on the objects in the tablespace. The tablespace is online while in the read-only state. You cannot make the `SYSTEM` or `SYSAUX` tablespaces read-only.
Note: The undo and temporary tablespaces cannot be made read-only.
- Offline:** You can take an online tablespace offline so that this portion of the database is temporarily unavailable for general use. The rest of the database is open and available for users to access data. When you take it offline, you can use the following options:
 - Normal: A tablespace can be taken offline normally if no error conditions exist for any of the data files of the tablespace. Oracle Database ensures that all data is written to disk by taking a checkpoint for all data files of the tablespace as it takes them offline.
 - Temporary: A tablespace can be taken offline temporarily even if there are error conditions for one or more files of the tablespace. Oracle Database takes the data files (which are not already offline) offline, performing checkpointing on them as it does so. If no files are offline, but you use the `Temporary` clause, media recovery is not required to bring the tablespace back online. However, if one or more files of the tablespace are offline because of write errors, and you take the tablespace offline temporarily, the tablespace requires recovery before you can bring it back online.

- Immediate: A tablespace can be taken offline immediately without Oracle Database taking a checkpoint on any of the data files. When you specify Immediate, media recovery for the tablespace is required before the tablespace can be brought online. You cannot take a tablespace offline immediately if the database is running in NOARCHIVELOG mode.

Note: System tablespaces may not be taken offline.

Changing the Size

You can add space to an existing tablespace by either adding data files to the tablespace or changing the size of an existing data file. You cannot add additional data files to bigfile tablespaces. You can make a tablespace either larger or smaller. However, you cannot make a data file smaller than the used space in the file. If you try to do so, you'll get an error.

Dropping Tablespaces

You can drop a tablespace and its contents (the segments contained in the tablespace) from the database if the tablespace and its contents are no longer required. You must have the `DROP TABLESPACE` system privilege to drop a tablespace.

When you drop a tablespace, the file pointers in the control file of the associated database are removed. If you are using Oracle Managed Files (OMF), the underlying operating system files are also removed. Otherwise, without OMF, you can optionally direct the Oracle server to delete the operating system files (data files) that constitute the dropped tablespace. If you do not direct the Oracle server to delete the data files at the same time that it deletes the tablespace, you must later use the appropriate commands of your operating system if you want them to be deleted.

You cannot drop a tablespace that contains segments with uncommitted updates from active transactions. For example, if a table in the tablespace is currently being used, or if the tablespace contains undo data that is needed to roll back uncommitted transactions, you cannot drop the tablespace. It is best to take the tablespace offline before dropping it.

Viewing Tablespace Information

Tablespace and data file information can be obtained by querying the following views:

- Tablespace information:
 - CDB_TABLESPACES and DBA_TABLESPACES
 - V\$TABLESPACE
- Data file information:
 - CDB_DATA_FILES and DBA_DATA_FILES
 - V\$DATAFILE
- Temp file information:
 - CDB_TEMP_FILES and DBA_TEMP_FILES
 - V\$TEMPFILE
- Tables in a tablespace:
 - ALL_TABLES



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Review: Implementing Oracle Managed Files (OMF)

- Specify file operations in terms of database objects rather than file names.

Parameter	Description
DB_CREATE_FILE_DEST	Defines the location of the default file system directory for data files and temporary files
DB_CREATE_ONLINE_LOG_DEST_n	Defines the location for redo log files and control file creation
DB_RECOVERY_FILE_DEST	Gives the default location for the fast recovery area

- Example:

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST='/u01/app/oracle/oradata';
SQL> CREATE TABLESPACE tbs_1;
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Managed Files

Oracle Managed Files (OMF) eliminates the need for you to directly manage the operating system files in an Oracle database. You specify operations in terms of database objects rather than file names. The database internally uses standard file system interfaces to create and delete files as needed for the following database structures:

- Tablespaces
- Redo log files
- Control files
- Archived logs
- Block change tracking files
- Flashback logs
- RMAN backups

A database can have a mixture of Oracle-managed and Oracle-unmanaged files. The file system directory specified by either of these parameters must already exist; the database does not create it. The directory must also have permissions for the database to create the files in it.

The table in the slide lists three initialization parameters that are used with OMF. The example in the slide shows that after the DB_CREATE_FILE_DEST initialization parameter is set, you can omit the DATAFILE clause from the CREATE TABLESPACE statement. The data file is created in the location specified by DB_CREATE_FILE_DEST.

In Oracle Database Cloud Service, OMF is enabled by default.

Naming Formats

Oracle-managed files have a specific naming format. For example, on Linux- and UNIX-based systems, the following format is used:

```
<destination_prefix>/o1_mf_%t_%u_.dbf
```

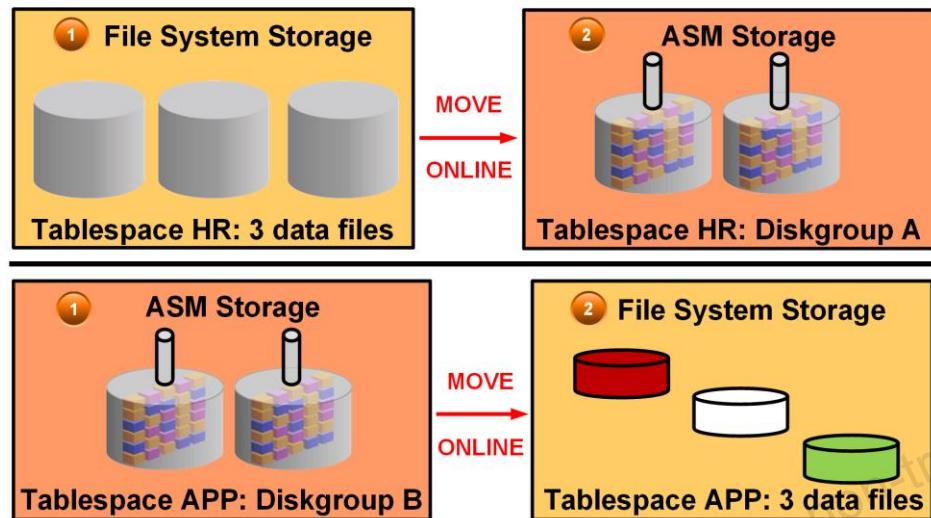
Do not rename an Oracle-managed file. The database identifies an Oracle-managed file based on its name. If you rename the file, the database is no longer able to recognize it as an Oracle-managed file and will not manage the file accordingly.

File Size

By default, Oracle-managed data files, including those for the `SYSTEM` and `SYSAUX` tablespaces, are 100 MB and auto-extensible.

Note: By default, Automatic Storage Management (ASM) uses OMF files, but if you specify an alias name for an ASM data file at tablespace creation time or when adding an ASM data file to an existing tablespace, then that file will not be OMF.

Moving or Renaming Online Data Files



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can rename and move an online data file from one kind of storage system to another while the database is open and accessing the file. The first example in the diagram illustrates moving the `HR` tablespace (three data files) from file system storage to ASM storage (diskgroup A). The second example illustrates moving the `APP` tablespace from ASM storage (diskgroup B) to file system storage (three data files).

Queries and DML and DDL operations can be performed while the data file is being moved. For example:

- `SELECT` statements against tables and partitions
- Creation of tables and indexes
- Rebuilding of indexes

Other notes:

- If objects are compressed while the data file is moved, the compression remains the same.
- You do not have to shut down the database or take the data file offline while you move a data file to another location, disk, or storage system.
- You can omit the `TO` clause only when an Oracle-managed file is used. In this case, the `DB_CREATE_FILE_DEST` initialization parameter should be set to indicate the new location.
- If the `REUSE` option is specified, the existing file is overwritten.
- If the `KEEP` clause is specified, the old file will be kept after the move operation. The `KEEP` clause is not allowed if the source file is an Oracle-managed file.
- Use the `V$SESSION_LONGOPS` view to display ongoing online move operations. Each ongoing operation has one row. The state transition of a successful online move operation is usually `NORMAL` to `COPYING` to `SUCCESS` and finally to `NORMAL`.

Examples: Moving and Renaming Online Data Files

- Relocating an online data file:

```
SQL> ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf'  
2 TO '/disk2/myexample01.dbf';
```

- Copying a data file from a file system to Automatic Storage Management (ASM):

```
SQL> ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf'  
2 TO '+DiskGroup2' KEEP;
```

- Renaming an online data file:

```
SQL> ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf'  
2 TO '/disk1/myexample02.dbf';
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You do not have to shut down the database or take the data file offline while you move a data file to another location, disk, or storage system.

The `TO` clause can be omitted only when an Oracle-managed file is used. In this case, the `DB_CREATE_FILE_DEST` parameter should be set to indicate the new location.

If the `REUSE` option is specified, the existing file is overwritten.

If the `KEEP` clause is specified, the old file will be kept after the move operation. The `KEEP` clause is not allowed if the source file is an Oracle-managed file.

Use the `V$SESSION_LONGOPS` view to display ongoing online move operations. Each ongoing operation has one row. The state transition of a successful online move operation is usually `NORMAL` to `COPYING` to `SUCCESS` and finally to `NORMAL`.

Tablespace Encryption by Default in DBCS



- In Oracle Database Cloud Service, user-created tablespaces are encrypted by default.
- Tablespaces created when the database is first created (in the root container, PDB seed, and PDB1) are NOT encrypted.
- The default encryption algorithm is AES128.
- The underlying architecture supporting this feature is Transparent Data Encryption (TDE).

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

All new tablespaces you create in a Database Cloud Service database are encrypted by default. In an Oracle Database 12c database, the tablespaces in the root (`CDB$ROOT`), the seed (`PDB$SEED`), and the PDB that is created when the database is created are not encrypted.

By default, any new tablespaces you create by using the SQL `CREATE TABLESPACE` command or any tool executing the `CREATE TABLESPACE` command are encrypted with the AES128 encryption algorithm. You do not need to include the `USING 'encrypt_algorithm'` clause to use the default encryption.

Controlling Tablespace Encryption by Default

Parameter Value	Description
ALWAYS	Any tablespace created will be transparently encrypted with the AES128 algorithm unless a different algorithm is specified in the ENCRYPTION clause.
CLOUD_ONLY (Default value)	Tablespaces created in a Database Cloud Service database will be transparently encrypted with the AES128 algorithm unless a different algorithm is specified in the ENCRYPTION clause. For non-Database Cloud Service databases, tablespaces will only be encrypted if the ENCRYPTION clause is specified.
DDL	Tablespaces are not transparently encrypted and are only encrypted if the ENCRYPTION clause is specified.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The ENCRYPT_NEW_TABLESPACES initialization parameter controls default encryption of new tablespaces. In Database Cloud Service databases, this parameter is set to CLOUD_ONLY and should not be modified.

Managing the Software Keystore and Master Encryption Key



- When the Database Cloud Service instance is created, a local auto-login software keystore is also created.
- The keystore is local to the compute node and is protected by a system-generated password.
- The auto-login software keystore is automatically opened when accessed.
- Use the `dbaascli` utility to change (rotate) the master encryption key.

```
DBAAS> tde rotate masterkey
Executing command tde rotate masterkey
Enter keystore password:
Successfully rotated TDE masterkey
```

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When the Database Cloud Service instance is created, a local auto-login software keystore is created. The keystore is local to the compute node and is protected by a system-generated password. The auto-login software keystore is automatically opened when accessed.

You can change (rotate) the master encryption key by using the `tde rotate masterkey` subcommand of the `dbaascli` utility. When you execute this subcommand, you are prompted for the keystore password. Enter the password specified when the service instance was created.

Creating an Encrypted Tablespace by Using a Nondefault Algorithm

- Use the ENCRYPTION USING clause to specify the algorithm:

```
SQL> CREATE TABLESPACE encrypt_ts
  DATAFILE '$ORACLE_HOME/dbs/encrypt.dat' SIZE 100M
  ENCRYPTION USING '3DES168'
  DEFAULT STORAGE (ENCRYPT);
```

- Restrictions:

- Temporary and undo tablespaces cannot be encrypted.
- BFILE data type and external tables are not encrypted.
- The key for an encrypted tablespace cannot be changed.
- The SYSTEM tablespace cannot be encrypted.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The CREATE TABLESPACE command has an ENCRYPTION clause that sets the encryption properties and an ENCRYPT storage parameter that causes the encryption to be used. You specify USING 'encrypt_algorithm' to indicate the name of the algorithm to be used. Valid algorithms are 3DES168, AES128, AES192, and AES256. The default is AES128.

In Oracle Database Cloud Service, tablespaces are encrypted by default by using the AES128 algorithm. Specify the ENCRYPTION USING clause if you want to use a different algorithm.

You can view the encryption properties in the V\$ENCRYPTED_TABLESPACES view.

Because tablespace encryption is performed at the I/O level, many of the restrictions that apply to TDE column encryption do not apply to tablespace encryption.

The following restrictions apply to tablespace encryption:

- Temporary and undo tablespaces cannot be encrypted. But when a data buffer containing data from an encrypted tablespace is written to an undo or temporary tablespace, that data block is encrypted.
- BFILE data type and external tables are not encrypted because they are not stored in tablespaces.
- Transportable tablespaces across different endian platforms are not supported.
- The key for encrypted tablespaces cannot be changed. A workaround is to create a tablespace with the desired properties and move all objects to the new tablespace.

Summary

In this lesson, you should have learned how to:

- Explain how table data is stored in the database
- Use SQL*Plus to:
 - Create and drop tablespaces
 - Alter tablespaces
 - View tablespace information
- Implement Oracle Managed Files (OMF)
- Use SQL*Plus to move and rename online data files
- Implement tablespace encryption



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 12: Overview

- 12-1: Viewing Tablespace Information
- 12-2: Creating a Tablespace
- 12-3: Creating a Tablespace that is Encrypted by Default



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Managing Storage Space

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe how the Oracle Database server automatically manages space
- Save space by using compression
- Proactively monitor and manage tablespace space usage
- Describe segment creation in the Oracle database
- Control deferred segment creation
- Reclaim wasted space from tables and indexes by using the segment shrink functionality
- Manage resumable space allocation



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Space Management Features

- Oracle Managed Files (OMF)
- Free-space management with bitmaps (“locally managed”) and automatic data file extension
- Proactive space management (default thresholds and server-generated alerts)
- Space reclamation (shrinking segments, online table redefinition)
- Capacity planning (growth reports)



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

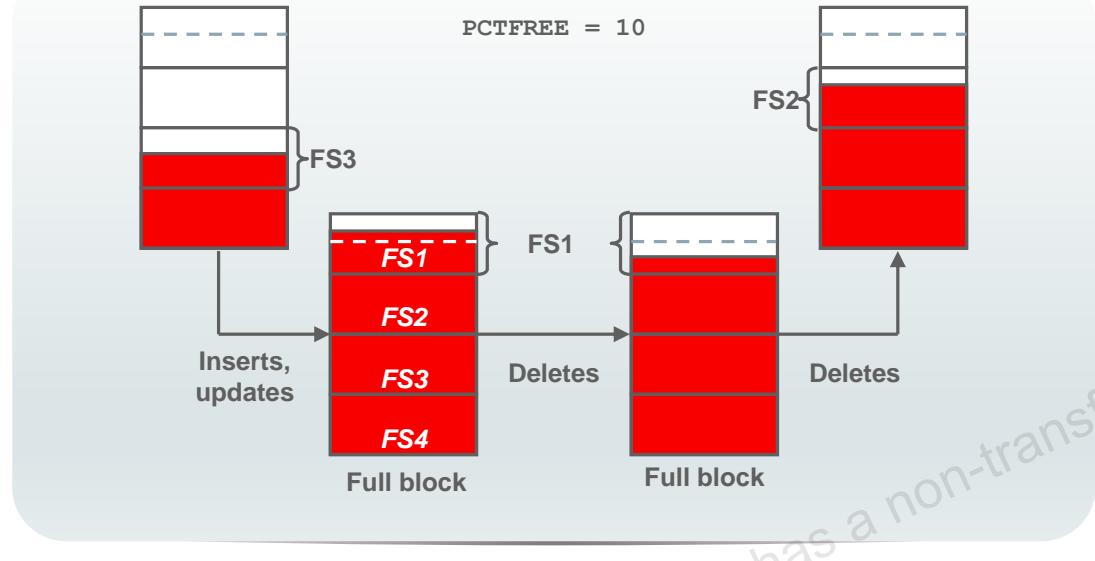
Space is automatically managed by the Oracle Database server. It generates alerts about potential problems and recommends possible solutions.

With Oracle Managed Files (OMF), you can specify operations in terms of database objects rather than file names. The Oracle Database server can manage free space within a tablespace with bitmaps. This is known as a “locally managed” tablespace. In addition, free space within segments located in locally managed tablespaces can be managed using bitmaps. This is known as Automatic Segment Space Management. The bitmapped implementation eliminates most space-related tuning of tables, while providing improved performance during peak loads. Additionally, the Oracle Database server provides automatic extension of data files, so the files can grow automatically based on the amount of data in the files.

When you create a database, proactive space monitoring is enabled by default (this causes no performance impact). The Oracle Database server monitors space utilization during normal space allocation and deallocation operations and alerts you if the free space availability falls below the predefined thresholds (which you can override). Advisors and wizards assist you with space reclamation.

For capacity planning, the Oracle Database server provides space estimates based on table structure and the number of rows and a growth trend report based on historical space utilization stored in the Automatic Workload Repository (AWR).

Block Space Management



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Space management involves the management of free space at the block level. With Automatic Segment Space Management, each block is divided into four sections, named FS1 (between 0 and 25% of free space), FS2 (25% to 50% free), FS3 (50% to 75% free), and FS4 (75% to 100% free).

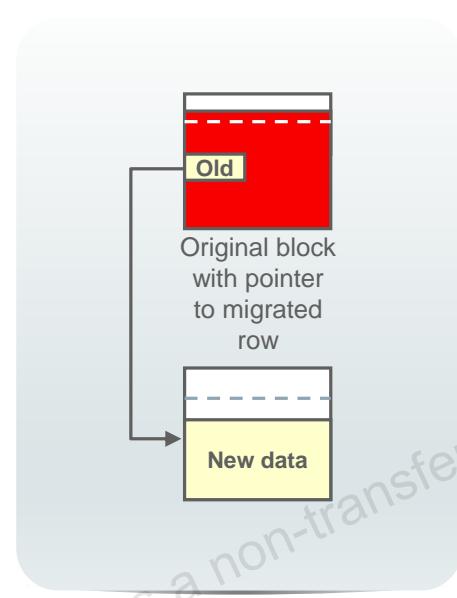
Depending on the level of free space in the block, its status is automatically updated. That way, depending on the length of an inserted row, you can tell whether a particular block can be used to satisfy an insert operation. Note that a “full” status means that a block is no longer available for inserts.

In the example in the slide, the block on the left is an FS3 block because it has between 50% and 75% free space. After some insert and update statements, PCTFREE is reached (the dashed line), and it is no longer possible to insert new rows in that block. The block is now considered as a “full” or FS1 block. The block is considered for insertion again as soon as its free space level drops below the next section. In the preceding case, it gets FS2 status as soon as the free space is more than 25%.

Note: Large object (LOB) data types (BLOB, CLOB, NCLOB, and BFILE) do not use the PCTFREE storage parameter. Uncompressed and OLTP-compressed blocks have a default PCTFREE value of 10; basic compressed blocks have a default PCTFREE value of 0.

Row Chaining and Migration

- On update: Row length increases, exceeding the available free space in the block.
- Data needs to be stored in a new block.
- Original physical identifier of row (`ROWID`) is preserved.
- The Oracle Database server needs to read two blocks to retrieve data.
- Segment Advisor finds segments containing the migrated rows.
- There is automatic coalescing of fragmented free space inside the block.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In two circumstances, the data for a row in a table may be too large to fit into a single data block. In the first case, the row is too large to fit into one data block when it is first inserted. In this case, the Oracle Database server stores the data for the row in a chain of data blocks (one or more) reserved for that segment. Row chaining most often occurs with large rows, such as rows that contain a column of data type `LONG` or `LONG RAW`. Row chaining in these cases is unavoidable.

However, in the second case, a row that originally fit into one data block is updated so that the overall row length increases, and the block's free space is already completely filled. In this case, the Oracle Database server migrates the data for the entire row to a new data block, assuming that the entire row can fit in a new block. The database preserves the original row piece of a migrated row to point to the new block containing the migrated row. The `ROWID` of a migrated row does not change.

When a row is chained or migrated, input/output (I/O) performance associated with this row decreases because the Oracle Database server must scan more than one data block to retrieve the information for the row.

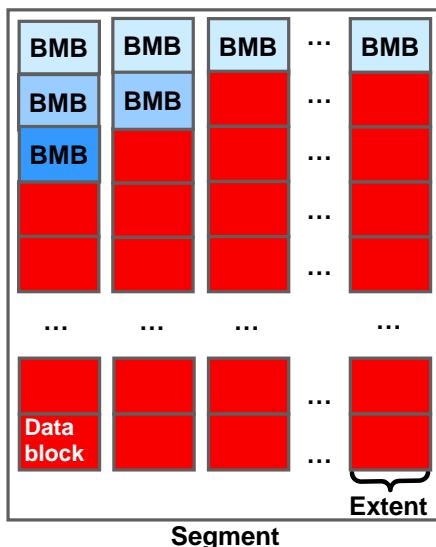
Segment Advisor finds the segments containing migrated rows that result from an `UPDATE`.

The Oracle Database server automatically and transparently coalesces the free space of a data block when:

- An `INSERT` or `UPDATE` statement attempts to use a block with sufficient free space for a new row piece
- The free space is fragmented so that the row piece cannot be inserted in a contiguous section of the block

After coalescing, the amount of free space is identical to the amount before the operation, but the space is now contiguous.

Free Space Management Within Segments



- Tracked by bitmaps in segments
- Benefits:
 - More flexible space utilization
 - Runtime adjustment
 - Multiple process search of bitmap blocks (BMBs)



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Free space can be managed automatically inside database segments. The in-segment free or used space is tracked with bitmaps. To take advantage of this feature, specify Automatic Segment Space Management when you create a locally managed tablespace. Your specification then applies to all segments subsequently created in this tablespace.

Automatic space management segments have a set of bitmap blocks (BMBs) describing the space utilization of the data blocks in that segment. BMBs are organized in a tree hierarchy. The root level of the hierarchy, which contains references to all intermediate BMBs, is stored in the segment header. The leaves of this hierarchy represent the space information for a set of contiguous data blocks that belong to the segment. The maximum number of levels inside this hierarchy is three.

Benefits of using automatic space management include:

- Better space utilization, especially for objects with highly varying row sizes
- Better runtime adjustment to variations in concurrent access
- Better multi-instance behavior in terms of performance or space utilization

Types of Segments

- A segment is a set of extents allocated for a certain logical structure.
- The different types of segments include:
 - Table and cluster
 - Index
 - Undo
 - Temporary
- Segments are dynamically allocated by the Oracle Database server.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Table and cluster segments: Each nonclustered table has a data segment. All table data is stored in the extents of the table segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.

Index segment: Each index has an index segment that stores all its data. For a partitioned index, each partition has an index segment.

Undo segment: Oracle Database maintains information to reverse changes made to the database. This information consists of records of the actions of transactions, collectively known as undo. Undo is stored in undo segments in an undo tablespace.

Temporary segment: A temporary segment is created by the Oracle Database server when a SQL statement needs a temporary database area to complete execution. When the statement finishes execution, the extents in the temporary segment are returned to the system for future use.

The Oracle Database server dynamically allocates space when the existing extents of a segment become full. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.

Allocating Extents

- Searching the data file's bitmap for the required number of adjacent free blocks
- Sizing extents with storage clauses:
 - UNIFORM
 - AUTOALLOCATE
- Viewing the extent map
- Obtaining deallocation advice



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

With locally managed tablespaces, the Oracle Database server looks for free space to allocate to a new extent by first determining a candidate data file in the tablespace and then searching the data file's bitmap for the required number of adjacent free blocks. If that data file does not have enough adjacent free space, then the Oracle Database server looks in another data file.

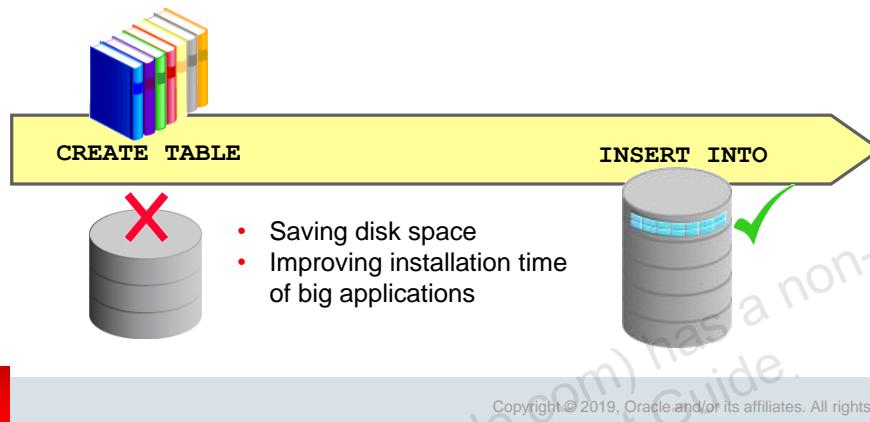
Two clauses affect the sizing of extents:

- With the **UNIFORM** clause, the database creates all extents of a uniform size that you specified (or a default size) for any objects created in the tablespace.
- With the **AUTOALLOCATE** clause, the database determines the extent-sizing policy for the tablespace.

The Oracle Database server provides a Segment Advisor that helps you determine whether an object has space available for reclamation on the basis of the level of space fragmentation within the object.

Understanding Deferred Segment Creation

- DEFERRED_SEGMENT_CREATION = TRUE is the default.
- Deferred segment is the default for tables, indexes, and partitions.
- Segment creation takes place as follows:
 - Table creation > Data dictionary operation
 - DML > Segment creation



When you create a nonpartitioned heap table, table segment creation is deferred to the first row insert. This functionality is enabled by default with the DEFERRED_SEGMENT_CREATION initialization parameter set to TRUE.

Advantages of this space allocation method:

- A significant amount of disk space can be saved for applications that create hundreds or thousands of tables upon installation, many of which might never be populated.
- The application installation time is reduced.

When you insert the first row into the table, the segments are created for the base table, its LOB columns, and its indexes. During segment creation, cursors on the table are invalidated. These operations have a small additional impact on performance.

With this allocation method, it is essential that you do proper capacity planning so that the database has enough disk space to handle segment creation when tables are populated.

Controlling Deferred Segment Creation

- With the DEFERRED_SEGMENT_CREATION parameter:
 - Initialization parameter file
 - ALTER SESSION command
 - ALTER SYSTEM command
- With the SEGMENT CREATION clause:
 - IMMEDIATE
 - DEFERRED (default)

```
CREATE TABLE SEG_TAB3(C1 number, C2 number)
SEGMENT CREATION IMMEDIATE TABLESPACE SEG_TBS;
CREATE TABLE SEG_TAB4(C1 number, C2 number)
SEGMENT CREATION DEFERRED;
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Segment creation can be controlled in two ways:

- With the DEFERRED_SEGMENT_CREATION initialization parameter set to TRUE or FALSE. This parameter can be set in the initialization parameter file. You can also control it via the ALTER SESSION or ALTER SYSTEM commands.
- With the SEGMENT CREATION clause of the CREATE TABLE command:
 - SEGMENT CREATION DEFERRED: If specified, segment creation is deferred until the first row is inserted into the table. This is the default behavior.
 - SEGMENT CREATION IMMEDIATE: If specified, segments are materialized during table creation.

This clause takes precedence over the DEFERRED_SEGMENT_CREATION parameter.

It is possible to force the creation of segments for an existing table with the ALTER TABLE ... MOVE command.

It is not possible to directly control the deferred segment creation for dependent objects such as indexes. They inherit this characteristic from their parent object—in this case, the table.

Restrictions and Exceptions

Segment creation on demand is not for the following:

- IOTs, clustered tables, or other special tables
- Tables in dictionary-managed tablespaces



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Segment creation on demand is not supported for IOTs, clustered tables, global temp tables, session-specific temp tables, internal tables, typed tables, AQ tables, SYS-owned tables, external tables, bitmap join indexes, and domain indexes. Tables owned by SYSTEM, PUBLIC, OUTLN, and XDB are also excluded.

Segment creation on demand is not supported for tables created in dictionary-managed tablespaces and for clustered tables. An attempt to do so creates segments.

If you create a table with deferred segment creation on a locally managed tablespace, it has no segments. If at a later time you migrate the tablespace to be dictionary-managed, any attempt to create segments produces errors. In this case, you must drop the table and re-create it.

Space-Saving Features

- No segments for unusable indexes and index partitions
- Creating an index without a segment:

```
CREATE INDEX test_i1 ON seg_test(c) UNUSABLE
```

- Removing any allocated space for an index:

```
ALTER INDEX test_i UNUSABLE
```

- Creating the segment for an index:

```
ALTER INDEX test_i REBUILD
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Additional features are implemented in Oracle Database to save space. All UNUSABLE indexes and index partitions are created without a segment. This functionality is completely transparent.

Example: If you have a table with three partitions and a local index, you see three table and three index segments when you query USER_SEGMENTS. If you execute the same query after you move one table partition to a new tablespace, you see three table segments and only two index segments because the unusable one is automatically deleted.

When you rebuild an index, you use an existing index as the data source. Rebuilding an index based on an existing data source removes intra-block fragmentation.

Private Temporary Tables

USER_PRIVATE_TEMP_TABLES

Private Temporary Tables (PTTs) exist only for the session that creates them.

- You can create a PTT with the CREATE PRIVATE TEMPORARY TABLE statement.
- The table name must start with ORA\$PTT_ :

PRIVATE_TEMP_TABLE_PREFIX = ORA\$PTT_

```
SQL> CREATE PRIVATE TEMPORARY TABLE ORA$PTT_mine (c1 DATE, ... c3 NUMBER(10,2));
```

- The CREATE PRIVATE TEMPORARY TABLE statement does not commit a transaction.
- Two concurrent sessions may have a PTT with the same name but different shape.



- PTT definition and contents are automatically dropped at the end of a session or transaction.

```
SQL> CREATE PRIVATE TEMPORARY TABLE ORA$PTT_mine (c1 DATE ...)
      ON COMMIT PRESERVE DEFINITION;
```

```
SQL> DROP TABLE ORA$PTT_mine;
```

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Private Temporary Tables (PTTs) are local to a specific session. In contrast with Global Temporary Tables, the definition and contents are local to the creating session only and are not visible to other sessions.

There are two types of durations for the created PTTs:

- **Transaction:** The PTT is automatically dropped when the transaction in which it was created ends with either a ROLLBACK or COMMIT. This is the default behavior if no ON COMMIT clause is defined at PTT creation.
- **Session:** The PTT is automatically dropped when the session that created it ends. This is the behavior if the ON COMMIT PRESERVE DEFINITION clause is defined at the PTT creation.

A PTT must be named with a prefix 'ORA\$PTT_'. The prefix is defined by default by the PRIVATE_TEMP_TABLE_PREFIX initialization parameter, modifiable at the instance level only.

Creating a PTT does not commit the current transaction. Because it is local to the current session, a concurrent session may also create a PTT with the same name but having a different shape.

At this time, PTTs cannot include User Defined Types, constraints, column default values, object types or XML types, or an identity clause.

PTTs must be created in the user schema. Creating a PTT in another schema, using the ALTER SESSION SET CURRENT SCHEMA command, is not allowed.

Table Compression: Overview

Reducing storage costs by compressing all data:

- Basic compression for direct-path insert operations: 10x
- Advanced row compression for all DML operations: 2–4x

Compression Method	Compression Ratio	CPU Overhead	CREATE and ALTER TABLE Syntax	Typical Applications
Basic table compression	High	Minimal	COMPRESS [BASIC]	DSS
Advanced row compression	High	Minimal	ROW STORE COMPRESS ADVANCED	OLTP, DSS



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Database supports three methods of table compression:

- Basic table compression
- Advanced row compression
- Hybrid columnar compression (with Exadata)

Oracle Corporation recommends compressing all data to reduce storage costs. The Oracle Database server can use table compression to eliminate duplicate values in a data block. For tables with highly redundant data, compression saves disk space and reduces memory use in the database buffer cache. Table compression is transparent to database applications.

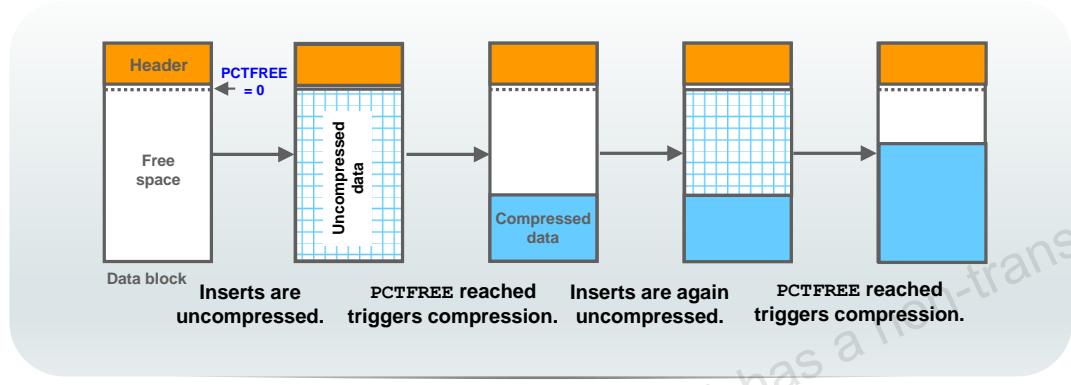
The `TABLE_COMPRESSION` clause is valid only for heap-organized tables. The `COMPRESS` keyword enables table compression. The `NOCOMPRESS` keyword disables table compression. `NOCOMPRESS` is the default.

With basic compression, the Oracle Database server compresses data at the time of performing bulk load by using operations such as direct loads or `CREATE TABLE AS SELECT`.

With `ROW STORE COMPRESS ADVANCED`, the Oracle Database server compresses data during all DML operations on the table.

Compression for Direct-Path Insert Operations

- Is enabled with CREATE TABLE ... COMPRESS BASIC
- Is recommended for bulk loading data warehouses
- Maximizes contiguous free space in blocks



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Enable basic table compression by using COMPRESS or COMPRESS BASIC. The Oracle Database server attempts to compress data during the following direct-path insert operations when it is productive to do so.

In earlier releases, this type of compression was called DSS table compression and was enabled using COMPRESS FOR DIRECT_LOAD OPERATIONS. This syntax has been deprecated.

Compression eliminates holes created due to deletions and maximizes contiguous free space in blocks.

The diagram in the slide shows you a data block evolution when that block is part of a compressed table. At the start, the block is empty and available for inserts. When you start inserting into this block, data is stored in an uncompressed format (as for uncompressed tables). However, as soon as the block is filled based on the PCTFREE setting of the block, the data is automatically compressed, potentially reducing the space it originally occupied.

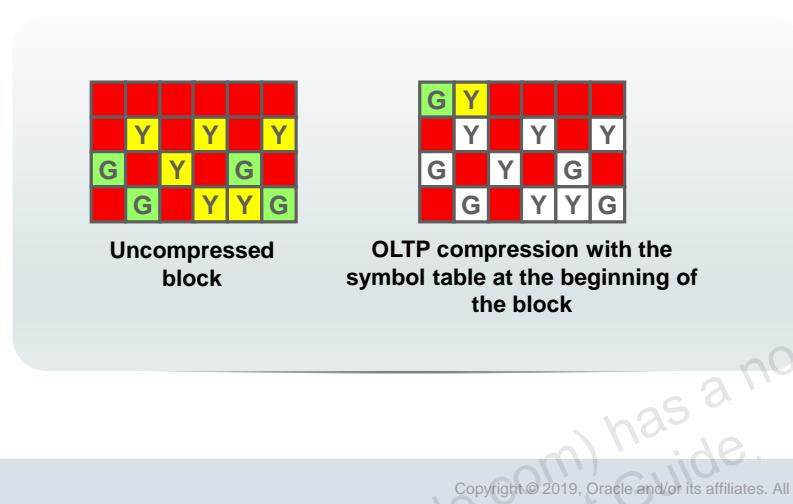
This allows for new uncompressed inserts to take place in the same block, until it is once again filled based on the PCTFREE setting. At that point, compression is triggered again to reduce the amount of space used in the block.

Note: Tables with COMPRESS or COMPRESS BASIC use a PCTFREE value of 0 to maximize compression, unless you explicitly set a value for the PCTFREE clause.

Tables with ROW STORE COMPRESS ADVANCED or NOCOMPRESS use the PCTFREE default value of 10 to maximize compression while still allowing for some future DML changes to the data, unless you override this default explicitly.

Advanced Row Compression for DML Operations

- Is enabled with CREATE TABLE ... ROW STORE COMPRESS ADVANCED
- Is recommended for active OLTP environments



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Enable advanced row compression by using ROW STORE COMPRESS ADVANCED.

The Oracle Database server compresses data during all DML operations on the table. This form of compression is recommended for active OLTP environments.

In earlier releases, OLTP table compression was enabled with COMPRESS FOR ALL OPERATIONS and COMPRESS FOR OLTP. This syntax has been deprecated.

With advanced row compression, duplicate values in the rows and columns in a data block are stored once at the beginning of the block in a symbol table. Duplicate values are replaced with a short reference to the symbol table, as shown in the diagram in the slide. Thus, information needed to re-create the uncompressed data is stored in the block.

To illustrate the principle of advanced row compression, the diagram shows two rectangles. The first red rectangle contains four small green squares labeled "G" and six yellow ones labeled "Y." They represent uncompressed blocks. At the beginning of the second red rectangle, there is only one green square labeled "G" and one yellow "Y" square, representing the symbol table. The second red diagram shows 10 white squares in the same position as the green and yellow ones. They are white because they are now only a reference, not consuming space for duplicate values.

Specifying Table Compression

- You can specify table compression for:
 - An entire heap-organized table
 - A partitioned table (each partition can have a different type or level of compression)
 - The storage of a nested table
- You cannot:
 - Specify basic and advanced row compression on tables with more than 255 columns
 - Drop a column if a table is compressed for direct loads, but you can drop it if the table is advance row compressed



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Table compression has the following restrictions:

- ROW STORE COMPRESS ADVANCED and COMPRESS BASIC are not supported for tables with more than 255 columns.
- You cannot drop a column from a table that is compressed for direct-load operations, although you can set such a column as unused. All the operations of the ALTER TABLE ... drop_column_clause are valid for tables that are compressed for OLTP.

Using Compression Advisor

- Analyzes objects to give an estimate of space savings for different compression methods
- Helps in deciding the correct compression level for an application
- Recommends various strategies for compression
 - Picks the right compression algorithm for a particular data set
 - Sorts on a particular column for increasing the compression ratio
 - Presents tradeoffs between different compression algorithms



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Compression Advisor analyzes database objects and determines the expected compression ratios that can be achieved for each compression level. It helps you determine the proper compression levels for your application and recommends various strategies for compression.

A compression advisor, provided by the `DBMS_COMPRESSION` package, helps you determine the compression ratio that can be expected for a specified table. The advisor analyzes the objects in the database, discovers the possible compression ratios that could be achieved, and recommends optimal compression levels. In addition to the `DBMS_COMPRESSION` package, compression advisor can also be used within the existing advisor framework (with the `DBMS_ADVISOR` package).

Resolving Space Usage Issues



- Resolve space usage issues by:
 - Adding or resizing data files
 - Setting AUTOEXTEND to ON
 - Shrinking objects
 - Reducing UNDO_RETENTION
- Check for long-running queries in temporary tablespaces.

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Tablespace thresholds are defined either as full or as available space in the tablespace. Critical and warning thresholds are the two thresholds that apply to a tablespace. The DBMS_SERVER_ALERT package contains procedures to set and get the threshold values. When the tablespace limits are reached, an appropriate alert is raised. The threshold is expressed in terms of a percentage of the tablespace size or in remaining bytes free. It is calculated in memory. You can have both a percentage and a byte-based threshold defined for a tablespace. Either or both of them may generate an alert.

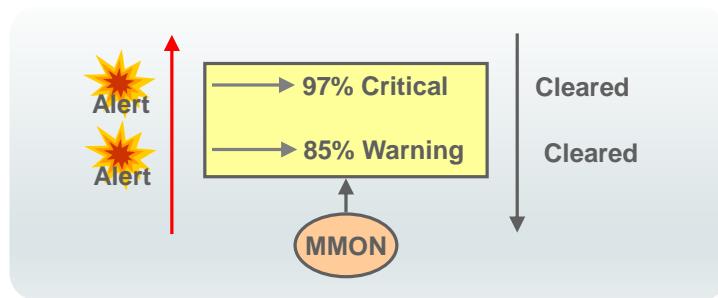
The ideal setting for the warning threshold trigger value results in an alert that is early enough to ensure that there is enough time to resolve the problem before it becomes critical, but late enough so that you are not bothered when space is not a problem.

The alert indicates that the problem can be resolved by doing one or more of the following:

- Adding more space to the tablespace by adding a file or resizing existing files or making an existing file auto-extensible
- Freeing up space on disks that contain any auto-extensible files
- Shrinking sparse objects in the tablespace

The diagram in the slide shows that the DBA receives a critical alert when the tablespace is 97% full and a warning when the tablespace is 85% full.

Monitoring Tablespace Space Usage



- Read-only and offline tablespaces: Do not set up alerts.
- Temporary tablespace: Threshold corresponds to space currently used by sessions.
- Undo tablespace: Threshold corresponds to space used by active and unexpired extents.
- Auto-extensible files: Threshold is based on the maximum file size.

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The database server tracks space utilization while performing regular space management activities. This information is aggregated by the MMON process. An alert is triggered when the threshold for a tablespace has been reached or cleared.

Alerts should not be flagged on tablespaces that are in read-only mode, or tablespaces that were taken offline, because there is not much to do for them.

In temporary tablespaces, the threshold value has to be defined as a limit on the used space in the tablespace.

For undo tablespaces, an extent is reusable if it does not contain active or unexpired undo. For the computation of threshold violation, the sum of active and unexpired extents is considered as used space.

For tablespaces with auto-extensible files, the thresholds are computed according to the maximum file size you specified or the maximum OS file size.

The diagram in the slide shows that the MMON process aggregates space utilization information and generates a critical alert when the tablespace is 97% full and a warning when it is 85% full. The alert is cleared after the space usage problem is addressed.

Reclaiming Space by Shrinking Segments

- Shrink is an online and in-place operation.
- It is applicable only to segments residing in ASSM tablespaces.
- Candidate segment types:
 - Heap-organized tables and index-organized tables
 - Indexes
 - Partitions and subpartitions
 - Materialized views and materialized view logs



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

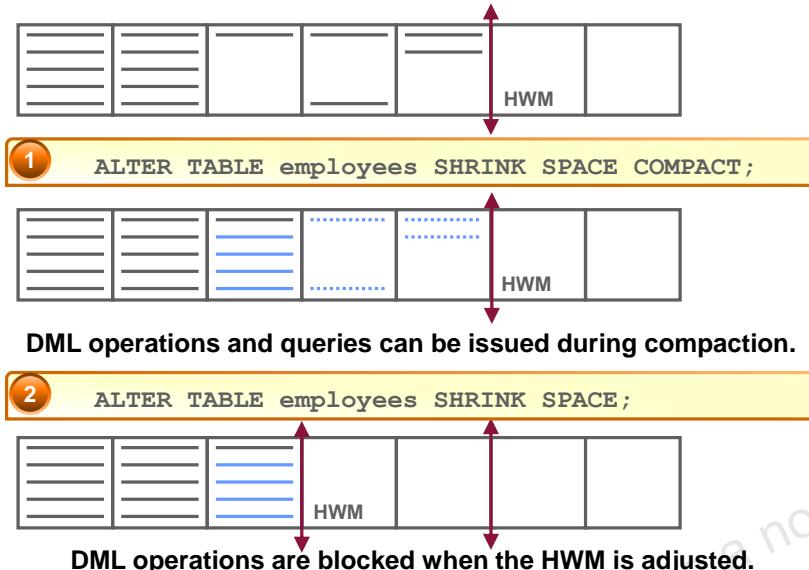
A shrink operation is an online and in-place operation because it does not need extra database space to be executed.

You cannot execute a shrink operation on segments managed by free lists. Segments in automatic segment space-managed tablespaces can be shrunk. However, the following objects stored in ASSM tablespaces cannot be shrunk:

- Tables in clusters
- Tables with `LONG` columns
- Tables with on-commit materialized views
- Tables with `ROWID`-based materialized views
- IOT mapping tables
- Tables with function-based indexes

Because a shrink operation may cause `ROWIDs` to change in heap-organized segments, you must enable row movement on the corresponding segment before executing a shrink operation on that segment. Row movement by default is disabled at segment level.

Shrinking Segments



ORACLE

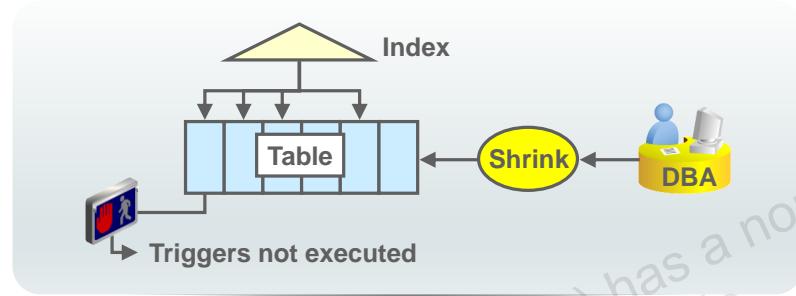
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide describes the two phases of a table shrink operation. Compaction is performed in the first phase. During this phase, rows are moved to the left part of the segment as much as possible. Internally, rows are moved by packets to avoid locking issues. After the rows have been moved, the second phase of the shrink operation is started. During this phase, the high-water mark (HWM) is adjusted and the unused space is released.

The `COMPACT` clause is useful if you have long-running queries that might span the shrink operation and attempt to read from blocks that have been reclaimed. When you specify the `SHRINK SPACE COMPACT` clause, the progress of the shrink operation is saved in the bitmap blocks of the corresponding segment. This means that the next time a shrink operation is executed on the same segment, the Oracle Database server remembers what has already been done. You can then reissue the `SHRINK SPACE` clause without the `COMPACT` clause during off-peak hours to complete the second phase.

Results of a Shrink Operation

- Improved performance and space utilization
- Indexes maintained
- Triggers not executed
- Number of migrated rows may be reduced
- Rebuilding secondary indexes on IOTs recommended



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Shrinking a sparsely populated segment improves the performance of scan and DML operations on that segment. This is because there are fewer blocks to look at after the segment has been shrunk. This is especially true for:

- Full table scans (fewer and denser blocks)
- Better index access (fewer I/Os on range ROWID scans due to a more compact tree)

Also, by shrinking sparsely populated segments, you enhance the efficiency of space utilization inside your database because more free space is made available for objects in need.

Index dependency is taken care of during the segment shrink operation. The indexes are in a usable state after shrinking the corresponding table. Therefore, no further maintenance is needed.

The actual shrink operation is handled internally as an `INSERT/DELETE` operation. However, DML triggers are not executed because the data itself is not changed.

As a result of a segment shrink operation, it is possible that the number of migrated rows is reduced. However, you should not always depend on reducing the number of migrated rows after a segment has been shrunk. This is because a segment shrink operation may not touch all the blocks in the segment. Therefore, it is not guaranteed that all the migrated rows are handled.

Note: It is recommended to rebuild secondary indexes on an index-organized table (IOT) after a shrink operation.

Managing Resumable Space Allocation

A resumable statement:

- Enables you to suspend large operations instead of receiving an error
- Gives you a chance to fix the problem while the operation is suspended, rather than starting over
- Is suspended for the following conditions:
 - Out of space
 - Maximum extents reached
 - Space quota exceeded
- Can be suspended and resumed multiple times



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Oracle Database server provides a means for suspending, and later resuming, the execution of large database operations in the event of space allocation failures. This enables you to take corrective action instead of the Oracle Database server returning an error to the user. After the error condition is corrected, the suspended operation automatically resumes. This feature is called “resumable space allocation.” The statements that are affected are called “resumable statements.” A statement executes in resumable mode only when the resumable statement feature has been enabled for the system or session.

Suspending a statement automatically results in suspending the transaction. Thus, all transactional resources are held through the suspension and resuming of a SQL statement. When the error condition disappears (for example, as a result of user intervention or perhaps sort space released by other queries), the suspended statement automatically resumes execution. A resumable statement is suspended when one of the following conditions occur:

- Out of space condition
- Maximum extents reached condition
- Space quota exceeded condition

A suspension timeout interval is associated with resumable statements. A resumable statement that is suspended for the timeout interval (the default is 2 hours) reactivates itself and returns the exception to the user. A resumable statement can be suspended and resumed multiple times.

Note: A maximum extents reached error happens only with dictionary-managed tablespaces.

Using Resumable Space Allocation

- Queries, DML operations, and certain DDL operations can be resumed if they encounter an out-of-space error.
- A resumable statement can be issued through SQL, PL/SQL, SQL*Loader, and Data Pump utilities, or Oracle Call Interface (OCI).
- A statement executes in resumable mode only if its session has been enabled by one of the following actions:
 - The `RESUMABLE_TIMEOUT` initialization parameter is set to a nonzero value.
 - An `ALTER SESSION ENABLE RESUMABLE` statement is issued.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Resumable space allocation is possible only when statements are executed within a session that has resumable mode enabled. There are two means of enabling and disabling resumable space allocation:

- Issue the `ALTER SESSION ENABLE RESUMABLE` command.
- Set the `RESUMABLE_TIMEOUT` initialization parameter to a nonzero value with an `ALTER SESSION` or `ALTER SYSTEM` statement.

When enabling resumable mode for a session or the database, you can specify a timeout period, after which a suspended statement errors out if no intervention has taken place. The `RESUMABLE_TIMEOUT` initialization parameter indicates the number of seconds before a timeout occurs. You can also specify the timeout period with the following command:

```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600
```

The value of `TIMEOUT` remains in effect until it is changed by another `ALTER SESSION ENABLE RESUMABLE` statement, it is changed by another means, or the session ends. The default timeout interval when using the `ENABLE RESUMABLE TIMEOUT` clause to enable resumable mode is 7,200 seconds or 2 hours.

You can also give a name to resumable statements. Example:

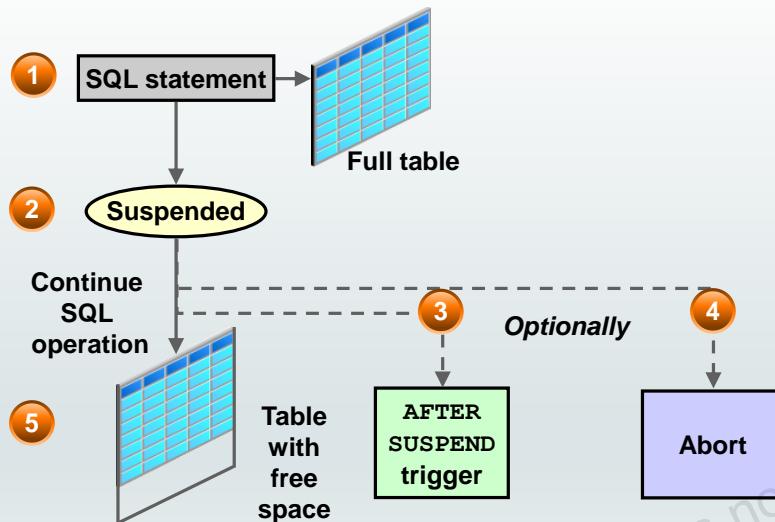
```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600 NAME 'multitab insert'
```

The name of the statement is used to identify the resumable statement in the `DBA_RESUMABLE` and `USER_RESUMABLE` views.

To automatically configure resumable statement settings for individual sessions, you can create and register a database-level LOGON trigger that alters a user's session. The trigger issues commands to enable resumable statements for the session, specifies a timeout period, and associates a name with the resumable statements issued by the session.

Because suspended statements can hold up some system resources, users must be granted the RESUMABLE system privilege before they are allowed to enable resumable space allocation and execute resumable statements.

Resuming Suspended Statements



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When a resumable statement is suspended, the error is not raised to the client. In order for corrective action to be taken, the Oracle Database server provides alternative methods for notifying users of the error and providing information about the circumstances.

The diagram in the slide illustrates the following example:

1. An `INSERT` statement encounters an error saying the table is full.
2. The `INSERT` statement is suspended, and no error is passed to the client.
3. Optionally, an `AFTER SUSPEND` trigger is executed.
4. Optionally, the `SQLERROR` exception is activated to abort the statement.
5. If the statement is not aborted and free space is successfully added to the table, the `INSERT` statement resumes execution.

Possible Actions During Suspension

When a resumable statement encounters a correctable error, the system internally generates the `AFTER SUSPEND` system event. Users can register triggers for this event at both the database and schema level. If a user registers a trigger to handle this system event, the trigger is executed after a SQL statement has been suspended. SQL statements executed within an `AFTER SUSPEND` trigger are always nonresumable and autonomous. Transactions started within the trigger use the `SYSTEM` rollback segment. These conditions are imposed to overcome deadlocks and reduce the chance of the trigger experiencing the same error condition as the statement.

Within the trigger code, you can use the `USER_RESUMABLE` or `DBA_RESUMABLE` views or the `DBMS_RESUMABLE.SPACE_ERROR_INFO` function to get information about the resumable statements.

When a resumable statement is suspended:

- The session invoking the statement is put into a wait state. A row is inserted into V\$SESSION_WAIT for the session with the EVENT column containing “statement suspended, wait error to be cleared.”
- An operation-suspended alert is issued on the object that needs additional resources for the suspended statement to complete.

Ending a Suspended Statement

When the error condition is resolved (for example, as a result of DBA intervention or perhaps sort space released by other queries), the suspended statement automatically resumes execution and the “resumable session suspended” alert is cleared.

A suspended statement can be forced to activate the SERVERERROR exception by using the DBMS_RESUMABLE.ABORT() procedure. This procedure can be called by a DBA or by the user who issued the statement. If the suspension timeout interval associated with the resumable statement is reached, the statement aborts automatically, and an error is returned to the user.

What Operations Are Resumable?

The following operations are resumable:

- **Queries:** SELECT statements that run out of temporary space (for sort areas)
- **DML:** INSERT, UPDATE, and DELETE statements
- The following DDL statements:
 - CREATE TABLE ... AS SELECT
 - CREATE INDEX
 - ALTER INDEX ... REBUILD
 - ALTER TABLE ... MOVE PARTITION
 - ALTER TABLE ... SPLIT PARTITION
 - ALTER INDEX ... REBUILD PARTITION
 - ALTER INDEX ... SPLIT PARTITION
 - CREATE MATERIALIZED VIEW



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The following operations are resumable:

- **Queries:** SELECT statements that run out of temporary space (for sort areas) are candidates for resumable execution. When using OCI, the OCISqlExecute() and OCISqlFetch() calls are candidates.
- **DML:** INSERT, UPDATE, and DELETE statements are candidates. The interface used to execute them does not matter; it can be OCI, SQLJ, PL/SQL, or another interface. Also, INSERT INTO...SELECT from external tables can be resumable.
- **DDL:** The following statements are candidates for resumable execution:
 - CREATE TABLE ... AS SELECT
 - CREATE INDEX
 - ALTER INDEX ... REBUILD
 - ALTER TABLE ... MOVE PARTITION
 - ALTER TABLE ... SPLIT PARTITION
 - ALTER INDEX ... REBUILD PARTITION
 - ALTER INDEX ... SPLIT PARTITION
 - CREATE MATERIALIZED VIEW

Summary

In this lesson, you should have learned how to:

- Describe how the Oracle Database server automatically manages space
- Save space by using compression
- Proactively monitor and manage tablespace space usage
- Describe segment creation in the Oracle database
- Control deferred segment creation
- Reclaim wasted space from tables and indexes by using the segment shrink functionality
- Manage resumable space allocation



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 13: Overview

- 13-1: Managing Tablespace Space
- 13-2: Using Compression
- 13-3: Handling Resumable Space Allocation



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Managing Undo Data

The ORACLE logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Explain DML and undo data generation
- Monitor and administer undo data
- Describe the difference between undo data and redo data
- Configure undo retention
- Guarantee undo retention
- Enable temporary undo
- Use the Undo Advisor



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Undo Data: Overview

Undo data is:

- A record of the action of a transaction
- Captured for every transaction that changes data
- Retained at least until the transaction is ended
- Used to support:
 - Rollback operations
 - Read-consistent queries
 - Oracle Flashback Query, Oracle Flashback Transaction, and Oracle Flashback Table
 - Recovery from failed transactions



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Oracle Database server saves the old value (undo data) when a process changes data in a database. It stores the data as it exists before modifications. Capturing undo data enables you to roll back your uncommitted data. Undo supports read-consistent and flashback queries. Undo can also be used to “rewind” (flash back) transactions and tables.

Read-consistent queries provide results that are consistent with the data as of the time a query started. For a read-consistent query to succeed, the original information must still exist as undo information. If the original data is no longer available, you receive a “Snapshot too old” error (ORA-01555). As long as the undo information is retained, the Oracle Database server can reconstruct data to satisfy read-consistent queries.

Flashback queries purposely ask for a version of the data as it existed at some time in the past. As long as undo information for that past time still exists, flashback queries can complete successfully. Oracle Flashback Transaction uses undo to create compensating transactions, to back out a transaction and its dependent transactions. With Oracle Flashback Table, you can recover a table to a specific point in time.

Undo data is also used to recover from failed transactions. A failed transaction occurs when a user session ends abnormally (possibly because of network errors or a failure on the client computer) before the user decides to commit or roll back the transaction. Failed transactions may also occur when the instance crashes or you issue the SHUTDOWN ABORT command.

In case of a failed transaction, the safest behavior is chosen, and the Oracle Database server reverses all changes made by a user, thereby restoring the original data.

Undo information is retained for all transactions, at least until the transaction is ended by one of the following:

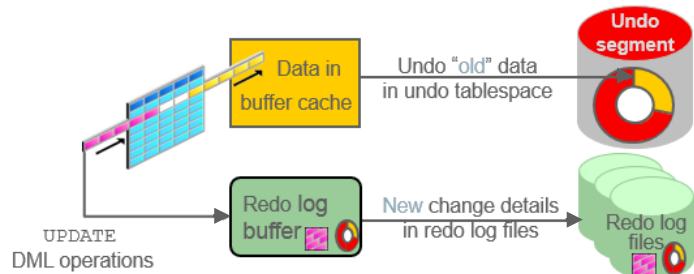
- User undoes a transaction (transaction rolls back)
- User ends a transaction (transaction commits)
- User executes a DDL statement, such as a CREATE, DROP, RENAME, or ALTER statement. If the current transaction contains any DML statements, the database server first commits the transaction and then executes and commits the DDL as a new transaction.
- User session terminates abnormally (transaction rolls back)
- User session terminates normally with an exit (transaction commits)

The amount of undo data that is retained and the time for which it is retained depend on the amount of database activity and the database configuration.

Note: Oracle Flashback Transaction leverages the online redo logs to mine the appropriate undo SQL for execution. It only uses undo as an artificial time boundary, to determine a redo mining start time for the target transaction, if a transaction start time is not supplied in the flashback transaction invocation.

Transactions and Undo Data

- Each transaction is assigned to only one undo segment.
- An undo segment can service more than one transaction at a time.



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When a transaction starts, it is assigned to an **undo segment**. Throughout the life of the transaction, when data is changed, the original (before the change) values are copied into the undo segment. You can see which transactions are assigned to which undo segments by checking the `V$TRANSACTION` dynamic performance view.

Undo segments are specialized segments that are automatically created by the database server as needed to support transactions. Like all segments, undo segments are made up of extents, which, in turn, consist of data blocks. Undo segments automatically grow and shrink as needed, acting as a circular storage buffer for their assigned transactions.

Transactions fill extents in their undo segments until a transaction is completed or all space is consumed. If an extent fills up and more space is needed, the transaction acquires that space from the next extent in the segment. After all extents have been consumed, the transaction either wraps around back into the first extent or requests a new extent to be allocated to the undo segment.

Note: Parallel DML and DDL operations can actually cause a transaction to use more than one undo segment. To learn more about parallel DML execution, see *Oracle Database Administrator's Guide*.

Storing Undo Information

- Undo information is stored in undo segments, which are stored in an undo tablespace.
- Undo tablespaces:
 - Are used only for undo segments
 - Have special recovery considerations
 - May be associated with only a single instance
 - Require that only one of them be the current writable undo tablespace for a given instance at any given time



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Undo segments can exist only in a specialized form of tablespace called an undo tablespace. You cannot create other segment types, such as tables, in the undo tablespace.

The Database Configuration Assistant (DBCA) automatically creates a smallfile undo tablespace. You can also create a bigfile undo tablespace. However, in a high-volume online transaction processing (OLTP) environment with many short concurrent transactions, contention could occur on the file header. An undo tablespace, stored in multiple data files, can resolve this potential issue.

Although a database may have many undo tablespaces, only one of them at a time can be designated as the current undo tablespace for any instance in the database.

Undo segments are automatically created and always owned by `SYS`. Because the undo segments act as a circular buffer, each segment has a minimum of two extents. The default maximum number of extents depends on the database block size but is very high (32,765 for an 8 KB block size).

Undo tablespaces are permanent, locally managed tablespaces with automatic extent allocation. They are automatically managed by the database.

Because undo data is required to recover from failed transactions (such as those that may occur when an instance crashes), undo tablespaces can be recovered only while the instance is in the `MOUNT` state.

Comparing Undo Data and Redo Data

	Undo	Redo
Record of	How to undo a change	How to reproduce a change
Used for	Rollback, read consistency, flashback	Rolling forward of database changes
Stored in	Undo segments	Redo log files



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Undo data and redo data seem similar at first, but they serve different purposes. Undo data is needed if there is a need to undo a change, and this occurs for read consistency and rollback. Redo data is needed if there is a need to perform the changes again, in cases where they are lost for some reason. Undo block changes are also written to the redo log.

The process of committing entails a verification that the changes in the transaction have been written to the redo log file, which is persistent storage on the disk, as opposed to memory. In addition, the redo log file is typically multiplexed. As a result, there are multiple copies of the redo data on the disk. Although the changes may not yet have been written to the data files where the table's blocks are actually stored, writing to the persistent redo log is enough to guarantee consistency of the database.

Assume that a power outage occurs just before committed changes have been reflected in the data files. This situation does not cause a problem because the transaction has been committed. When the system starts up again, it is able to roll forward any redo records that are not yet reflected in data files at the time of the outage.

Managing Undo

- Automatic undo management:
 - Fully automated management of undo data and space in a dedicated undo tablespace
 - For all sessions
 - Self-tuning in AUTOEXTEND tablespaces to satisfy long-running queries
 - Self-tuning in fixed-size tablespaces for best retention
- DBA tasks in support of Flashback operations:
 - Configuring undo retention
 - Changing the undo tablespace to a fixed size
 - Avoiding space and “snapshot too old” errors



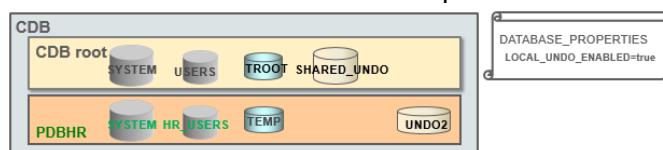
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Oracle Database server provides automatic undo management, which is a fully automated mechanism for managing undo information and space in a dedicated undo tablespace for all sessions. The system automatically tunes itself to provide the best possible retention of undo information. More precisely, the undo retention period for auto-extending tablespaces is tuned to be slightly longer than the longest-running active query. For fixed-size undo tablespaces, the database dynamically tunes for best possible retention.

Although, by default, the Oracle Database server manages undo data and space automatically, you may need to perform some tasks if your database is using Flashback operations. The administration of undo should prevent space errors, the use of too much space, and “Snapshot too old” errors.

Comparing SHARED Undo Mode and LOCAL Undo Mode

- There are two undo modes in the multitenant architecture: SHARED and LOCAL .
 - There is only one SHARED undo tablespace (in CDB root).
 - There can be a LOCAL undo tablespace in each PDB.
- When is LOCAL undo mode required?
 - Hot cloning
 - Near-zero down time PDB relocation



```
SQL> STARTUP UPGRADE;
SQL> ALTER DATABASE LOCAL UNDO ON;
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Using the LOCAL undo mode is required when cloning a PDB in hot mode, performing a near-zero down time PDB relocation, refreshing PDBs, or using proxy PDBs.

You can set a CDB in LOCAL undo mode either at CDB creation or by altering the CDB property.

When the database property LOCAL_UNDO_ENABLED is FALSE, which is the default, there is only one undo tablespace that is created in the CDB root, and that is shared by all containers.

When LOCAL_UNDO_ENABLED is TRUE, every container in the CDB uses LOCAL undo, and each PDB must have its own LOCAL undo tablespace. To maintain ease of management and provisioning, undo tablespace creation happens automatically and does not require any action from the user.

When a PDB is opened and an undo tablespace is not available, it is automatically created.

Configuring Undo Retention

- `UNDO_RETENTION` specifies (in seconds) how long already committed undo information is to be retained.
- Set this parameter when:
 - The undo tablespace has the `AUTOEXTEND` option enabled
 - You want to set undo retention for LOBs
 - You want to guarantee retention



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The `UNDO_RETENTION` initialization parameter specifies (in seconds) the low threshold value of undo retention. Set the minimum undo retention period for the auto-extending undo tablespace to be as long as the longest expected Flashback operation. For auto-extending undo tablespaces, the system retains undo for at least the time specified in this parameter and automatically tunes the undo retention period to meet the undo requirements of the queries. But this autotuned retention period may be insufficient for your Flashback operations.

For fixed-size undo tablespaces, the system automatically tunes for the best possible undo retention period on the basis of undo tablespace size and usage history; it ignores `UNDO_RETENTION` unless retention guarantee is enabled. So for automatic undo management, the `UNDO_RETENTION` setting is used for the three cases listed in the slide. In cases other than these three, this parameter is ignored.

Categories of Undo

Category	Description
Active: Uncommitted undo information	Supports an active transaction and is never overwritten
Unexpired: Committed undo information	Is required to meet the undo retention interval
Expired: Expired undo information	Overwritten when space is required for an active transaction



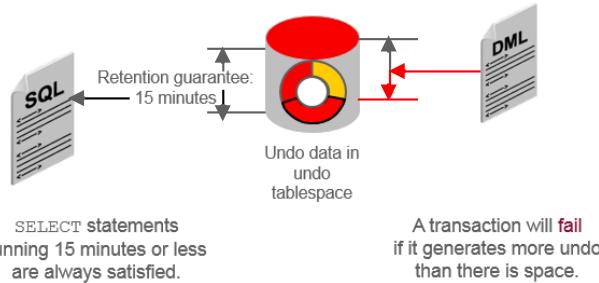
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Undo information is divided into three categories:

- **Uncommitted undo information (Active):** Supports a currently running transaction and is required if a user wants to roll back or if the transaction has failed. Uncommitted undo information is never overwritten.
- **Committed undo information (Unexpired):** Is no longer needed to support a running transaction but is still needed to meet the undo retention interval. It is also known as “unexpired” undo information. Committed undo information is retained when possible without causing an active transaction to fail because of lack of space.
- **Expired undo information (Expired):** Is no longer needed to support a running transaction. Expired undo information is overwritten when space is required by an active transaction.

Guaranteeing Undo Retention

```
SQL> ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
```



This example is based on an `UNDO_RETENTION` setting of 900 seconds (15 minutes).

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The default undo behavior is to overwrite the undo information of committed transactions that has not yet expired rather than to allow an active transaction to fail because of lack of undo space.

This behavior can be changed by guaranteeing retention. With guaranteed retention, undo retention settings are enforced even if they cause transactions to fail.

`RETENTION GUARANTEE` is a tablespace attribute rather than an initialization parameter. This attribute can be changed only with SQL command-line statements.

The syntax to change an undo tablespace to guarantee retention is:

```
SQL> ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
```

To return a guaranteed undo tablespace to its normal setting, use the following command:

```
SQL> ALTER TABLESPACE undotbs1 RETENTION NOGUARANTEE;
```

The retention guarantee applies only to undo tablespaces. Attempts to set it on a non-undo tablespace result in the following error:

```
SQL> ALTER TABLESPACE example RETENTION GUARANTEE;
```

ERROR at line 1:

```
ORA-30044: 'Retention' can only specified for undo tablespace.
```

Changing an Undo Tablespace to a Fixed Size

- Rationale:
 - Supporting Flashback operations
 - Limiting tablespace growth
- Steps:
 - Run the regular workload.
 - The self-tuning mechanism establishes the minimum required size.
 - (Optional) Use the Enterprise Manager Cloud Control Undo Advisor, which calculates the required size for future growth.
 - (Optional) Change the undo tablespace to a fixed size.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You might have two reasons for changing the undo tablespace to a fixed size: to support Flashback operations (where you expect future use of the undo) or to prevent the tablespace from growing too large.

If you decide to change the undo tablespace to a fixed size, you must choose a large enough size to avoid the following two errors:

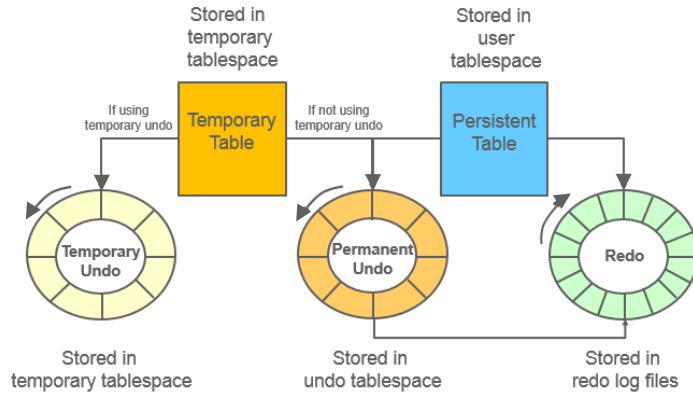
- DML failures (because there is not enough space to create the undo for new transactions)
- “Snapshot too old” errors (because there was insufficient undo data for read consistency)

Oracle recommends that you run a regular, full workload allowing the undo tablespace to grow to its minimum required size. The automatically gathered statistics include the duration of the longest-running query and the undo generation rate. Computing the minimum undo tablespace size based on these statistics is advisable for a system without Flashback operations and for a system for which you do not expect longer-running queries in the future.

You can use the Enterprise Manager Cloud Control Undo Advisor to enter your desired duration for the undo period for longer-running queries and flashback.

Note: For fixed-size undo tablespaces, the system automatically tunes for the maximum possible undo retention period, based on undo tablespace size and usage history, and ignores `UNDO_RETENTION` unless retention guarantee is enabled.

Temporary Undo: Overview



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Temporary tables are widely used as scratch areas for staging intermediate results. This is because changing those tables is much faster than with non-temporary tables. The performance gain is mainly because no redo entries are directly generated for changes on temporary tables. However, the undo for operations on temporary tables (and indexes) is still logged to the redo log.

Undo for temporary tables is useful for consistent reads and transaction rollbacks during the life of that temporary object. Beyond this scope, the undo is superfluous. Therefore, it need not be persisted in the redo stream. For instance, transaction recovery just discards undo for temporary objects.

Starting with Oracle Database 12c, it is possible for undo generated by temporary tables' transactions to be stored in a separate undo stream directly in the temporary tablespace to avoid that undo being logged in the redo stream. This mode is called temporary undo.

Note: A temporary undo segment is session private. It stores undo for the changes to temporary tables (temporary objects in general) belonging to the corresponding session.

Temporary Undo Benefits

- Reduces the amount of undo stored in the undo tablespaces
- Reduces the amount of redo data written to the redo log
- Enables DML operations on temporary tables in a physical standby database with the Oracle Active Data Guard option



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Enabling temporary undo provides the following benefits:

- Temporary undo reduces the amount of undo stored in the undo tablespaces. Less undo in the undo tablespaces can result in more realistic undo retention period requirements for undo records.
- Performance is improved because less data is written to the redo log, and components that parse redo log records, such as LogMiner, perform better because there is less redo data to parse.
- Temporary undo enables data manipulation language (DML) operations on temporary tables in a physical standby database with the Oracle Active Data Guard option. However, data definition language (DDL) operations that create temporary tables must be issued on the primary database.

Enabling Temporary Undo

- Enable temporary undo for a session:

```
SQL> ALTER SESSION SET temp_undo_enabled = true;
```

- Enable temporary undo for the database instance:

```
SQL> ALTER SYSTEM SET temp_undo_enabled = true;
```

- Temporary undo mode is selected when a session first uses a temporary object.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can enable temporary undo for a specific session or for the entire database. When you enable temporary undo for a session using an `ALTER SESSION` statement, the session creates temporary undo without affecting other sessions. When you enable temporary undo for the system using an `ALTER SYSTEM` statement, all existing sessions and new sessions create temporary undo.

When a session uses temporary objects for the first time, the current value of the `TEMP_UNDO_ENABLED` initialization parameter is set for the rest of the session. Therefore, if temporary undo is enabled for a session and the session uses temporary objects, then temporary undo cannot be disabled for the session. Similarly, if temporary undo is disabled for a session and the session uses temporary objects, then temporary undo cannot be enabled for the session.

The feature of temporary undo is available for databases with the `COMPATIBLE` initialization parameter set to at least `12.1.0.0.0`.

Note: Temporary undo is enabled by default for a physical standby database with the Oracle Active Data Guard option. The `TEMP_UNDO_ENABLED` initialization parameter has no effect on a physical standby database with the Active Data Guard option because of the default setting.

Monitoring Temporary Undo

```
SQL> SELECT to_char(BEGIN_TIME, 'dd/mm/yy hh24:mi:ss') "BEGIN TIME",
  2  txncount "TXNCNT", maxconcurrency, undoblkcnt, uscount "USCNT",
  3  nospaceerrcnt "NOSPEERRCNT"
  4  FROM  v$tempundostat;

BEGIN TIME          TXNCNT MAXCONCURRENCY UNDOBLKCNT USCNT NOSPEERRCNT
-----  -----
...
19/08/12 22:19:44      0            0           0       0       0
19/08/12 22:09:44      0            0           0       0       0
...
19/08/12 13:09:44      0            0           0       0       0
19/08/12 12:59:44      3            1           24      1       0
576 rows selected.
SQL>
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

V\$TEMPUNDOSTAT shows various statistics related to the temporary undo log for this database instance. It displays a histogram of statistical data to show how the system is working. Each row in the view keeps statistics collected in the instance for a 10-minute interval. The rows are in descending order of the BEGIN_TIME column value. This view contains a total of 576 rows, spanning a four-day cycle. This view is similar to the V\$UNDOSTAT view.

The example shows you some of the important columns of the V\$TEMPUNDOSTAT view:

- BEGIN_TIME: The beginning of the time interval
- TXNCOUNT: The total number of transactions that have bound to temp undo segment within the corresponding time interval
- MAXCONCURRENCY: The highest number of transactions executed concurrently, which modified temporary objects within the corresponding time interval
- UNDOBLKCNT: The total number of temporary undo blocks consumed during the corresponding time interval
- USCNT: The temp undo segments created during the corresponding time interval
- NOSPACEERRCNT: The total number of times the “no space left for temporary undo” error was raised during the corresponding time interval

Note: For more information on V\$TEMPUNDOSTAT, refer to *Oracle Database Reference Guide*.

Viewing Undo Information

The screenshot shows the Oracle Enterprise Manager Database Express interface. The top navigation bar includes links for Configuration, Storage, Security, and Performance. The main content area is titled "Undo Management Details".
Configuration Section:

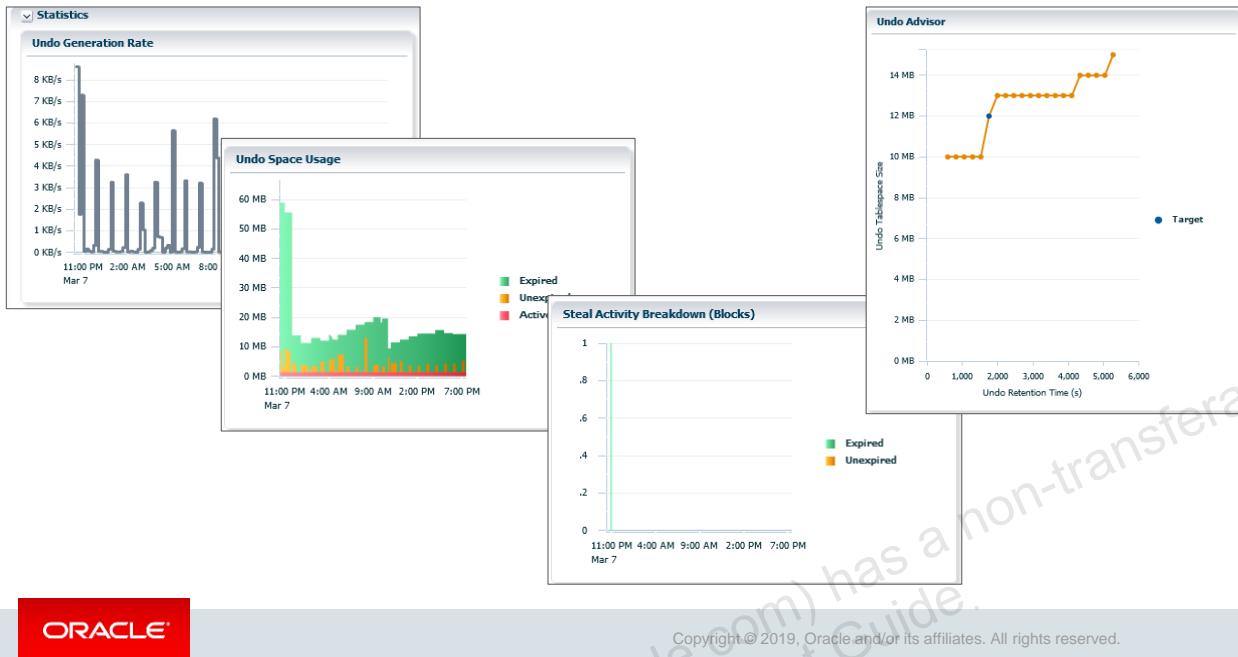
- Undo Summary:** Undo Management is set to auto, and the Low Undo Retention Threshold is 900s.
- Tablespace:** A single undo tablespace named UNDOTBS1 is listed, with a size of 60MB (74.5% free) and Auto Extensible set to Yes (maximum size unlimited).
- Errors and Warnings:** Shows 0 Snapshot Too Old Errors, 0 Out of Space Errors, and 0 Unexpired Blocks Stolen.
- Advisor Findings:** Both Health and Setting show "No problems".

Undo Statistics Summary Section:

- Analysis Period (Last Day):** The analysis period from Wednesday, March 7, 2018, 10:30:38 PM to Thursday, March 8, 2018, 7:28:31 PM had a duration of 20 hours, 57 minutes, 53 seconds. Required Undo Retention was 29 minutes, 17 seconds.
- Undo Retention Analysis:** Required Undo Retention was 29 minutes, 17 seconds, and Best Undo Retention was 234 days, 8 hours, 49 minutes, 12 seconds.
- Undo Statistics:** Undo Generation Rate was 829 B/s, Maximum Undo Used was 59MB, and the longest SQL was f3yfg5oga0r8n. Longest SQL Execution Time was 29 minutes, 17 seconds. Transaction Rate was 0 transaction(s) per second, and Maximum Concurrency was 8.

You can view undo information on the Undo Management Details page in Enterprise Manager Database Express. The screenshot shows the two regions (on the top left) of the Undo Management Details page while connected to the container database. The Undo Summary region provides details on undo settings, undo tablespace information, errors and warnings, and the Undo Advisor findings. The Undo Statistics Summary region provides details on the undo analysis period for the last day, output of the undo retention analysis, and undo statistics such as undo generation rate, maximum undo used, longest SQL execution time, transaction rate, and the maximum concurrency.

Viewing Undo Activity



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Oracle Enterprise Manager Database Express Undo Management Details page includes four additional graphs, as shown in the screenshot in the slide. They include:

- **Undo Generation Rate:** Displays the undo generation (in KB per second)
- **Undo Space Usage:** Shows the use of space in the tablespace with different colors for expired, unexpired, and active extents
- **Steal Activity Breakdown:** Shows the number of attempts to steal expired undo blocks from other undo segments and attempts to obtain undo space by stealing unexpired extents from other transactions
- **Undo Advisor:** Shows the ratio of undo retention times to tablespace sizing needs, along with the current target setting

Summary

In this lesson, you should have learned how to:

- Explain DML and undo data generation
- Monitor and administer undo data
- Describe the difference between undo data and redo data
- Configure undo retention
- Guarantee undo retention
- Enable temporary undo
- Use the Undo Advisor



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 14: Overview

- 14-1: Managing Undo Data



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Moving Data



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

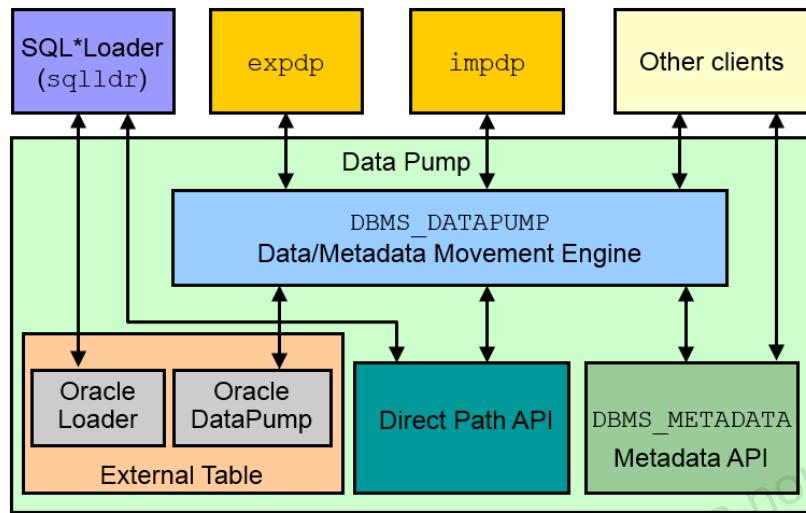
- Describe ways to move data
- Explain the general architecture of Oracle Data Pump
- Use Data Pump Export and Import to move data between Oracle databases
- Use SQL*Loader to load data from a non-Oracle database (or user files)
- Use external tables to move data via platform-independent files
- Describe methods that can be used to migrate databases to Oracle Database Cloud Service



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Moving Data: General Architecture



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Major functional components include:

- **DBMS_DATAPUMP**: It contains the API for high-speed export and import utilities for bulk data and metadata movement.
- **Direct Path API (DPAPI)**: Oracle Database supports a Direct Path API interface that minimizes data conversion and parsing at both unload and load time.
- **DBMS_METADATA**: Used by worker processes for all metadata unloading and loading. Database object definitions are stored by using XML rather than SQL.
- **External Table**: With the **ORACLE_DATAPUMP** and **ORACLE_LOADER** access drivers, you can store data in external tables (that is, in platform-independent files). The **SELECT** statement reads external tables as though they were stored in an Oracle database.
- **SQL*Loader**: It has been integrated with external tables, providing automatic migration of loader control files to external table access parameters.
- **expdp** and **impdp**: They are thin layers that make calls to the **DBMS_DATAPUMP** package to initiate and monitor Data Pump operations.
- **Other clients**: Applications (such as replication, transportable tablespaces, and user applications) that benefit from this infrastructure. **SQL*Plus** may also be used as a client of **DBMS_DATAPUMP** for simple status queries against ongoing operations.

Oracle Data Pump: Overview

As a server-based facility for high-speed data and metadata movement, Oracle Data Pump:

- Is callable via `DBMS_DATAPUMP`
- Provides the following tools:
 - `expdp` and `impdp`
 - GUI interface in Enterprise Manager Cloud Control
- Provides several data movement methods:
 - Conventional path load
 - Direct path
 - External tables
 - Transportable tablespace
 - Network link support
- Detaches from and reattaches to long-running jobs
- Restarts Data Pump jobs



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Data Pump enables very high-speed data and metadata loading and unloading of Oracle databases. The Data Pump infrastructure is callable via the `DBMS_DATAPUMP` PL/SQL package. Thus, custom data movement utilities can be built by using Data Pump.

Oracle Database provides the following tools:

- Command-line export and import clients called `expdp` and `impdp`, respectively
- An export and import interface in Enterprise Manager Cloud Control

Data Pump automatically decides the data access methods to use; these can be either direct path or external tables. Data Pump uses direct path load and unload when a table's structure allows it and when maximum single-stream performance is desired. However, if there are clustered tables, encrypted columns, or several other items, Data Pump uses external tables rather than direct path to move the data.

Conventional Path Load is used when Data Pump is not able to load data into a table by using either direct path or external tables. The ability to detach from and reattach to long-running jobs without affecting the job itself enables you to monitor jobs from multiple locations while they are running. All stopped Data Pump jobs can be restarted without loss of data as long as the metadata remains undisturbed. It does not matter whether the job is stopped voluntarily or involuntarily due to a crash.

Oracle Data Pump: Benefits

Data Pump offers many benefits and features, such as:

- Fine-grained object and data selection
- Explicit specification of database version
- Parallel execution
- Network mode in a distributed environment
- Remapping capabilities
- Data sampling and metadata compression
- Compression of data during a Data Pump export
- Security through encryption
- Ability to export XMLType data as CLOBs



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The EXCLUDE, INCLUDE, and CONTENT parameters are used for fine-grained object and data selection.

You can specify the database version for objects to be moved (using the VERSION parameter) to create a dump file set that is compatible with a previous release of Oracle Database that supports Data Pump.

You can use the PARALLEL parameter to specify the maximum number of threads of active execution servers operating on behalf of the export job.

Network mode enables you to export from a remote database directly to a dump file set. This can be done by using a database link to the source system.

During import, you can change the target data file names, schemas, and tablespaces:

- Rename tables during an import operation.
- Remap data as it is being imported into a new database.

In addition, you can specify a percentage of data to be sampled and unloaded from the source database when performing a Data Pump export. This can be done by specifying the SAMPLE parameter.

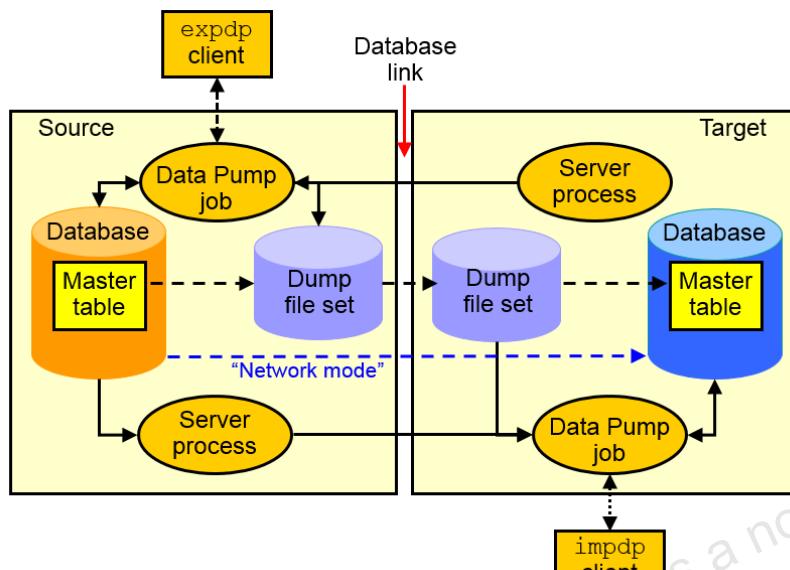
You can use the COMPRESSION parameter to indicate whether the metadata should be compressed in the export dump file so that it consumes less disk space. If you compress the metadata, it is automatically uncompressed during import. You can choose to compress both data and metadata, only data, only metadata, or no data during an export. This feature requires the Oracle Advanced Compression option.

You can also specify encryption options to encrypt. Encryption requires the Oracle Advanced Security option:

- Both data and metadata, only data, only metadata, no data, or only encrypted columns during an export
- With a particular encryption algorithm to use during an export
- Using the type of security to use for performing encryption and decryption during an export. For example, perhaps the dump file set will be imported into a different or remote database and must remain secure in transit. Or perhaps the dump file set will be imported on-site using the Oracle Encryption Wallet, but it may also need to be imported off-site where the Oracle Encryption Wallet is not available.

You can also define that XMLType columns are to be exported in uncompressed CLOB format regardless of the XMLType storage format that was defined for them.

Data Pump Export and Import Clients



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Data Pump Export is a utility for unloading data and metadata into a set of operating system files called dump file sets. Data Pump Import is used to load metadata and data stored in an export dump file set into a target system.

The Data Pump API accesses its files on the server rather than on the client.

These utilities can also be used to export from a remote database directly to a dump file set or to load the target database directly from a source database with no intervening files. This is known as network mode. This mode is particularly useful to export data from a read-only source database.

At the center of every Data Pump operation is the master table, which is a table created in the schema of the user running the Data Pump job. The master table maintains all aspects of the job. The master table is built during a file-based export job and is written to the dump file set as the last step. Conversely, loading the master table into the current user's schema is the first step of a file-based import operation and is used to sequence the creation of all objects imported.

Note: The master table is the key to Data Pump's restart capability in the event of a planned or unplanned stopping of the job. The master table is dropped when the Data Pump job finishes normally.

Data Pump Interfaces and Modes

- Data Pump Export and Import interfaces:
 - Command line
 - Parameter file
 - Interactive command line
 - Enterprise Manager Cloud Control
- Data Pump Export and Import modes:
 - Full
 - Schema
 - Table
 - Tablespace
 - Transportable tablespace
 - Transportable database



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can interact with Data Pump Export and Import by using one of the following interfaces:

- **Command-line interface:** Enables you to specify most of the export parameters directly on the command line.
- **Parameter file interface:** Enables you to specify all command-line parameters in a parameter file. The only exception is the PARFILE parameter.
- **Interactive-command interface:** Stops logging in to the terminal and displays the export or import prompts, where you can enter various commands. This mode is enabled by pressing Ctrl + C during an export operation that is started with the command-line interface or the parameter file interface. Interactive-command mode is also enabled when you attach to an executing or stopped job.
- **Oracle Enterprise Manager Cloud Control:** Select Schema > Database Export/Import. In the menu, select the export or import operation you want to execute.

Data Pump Export and Import provide different modes for unloading and loading different portions of the database. The mode is specified on the command line by using the appropriate parameter. The available modes are also listed in the diagram in the slide.

- FULL=YES: All data and metadata of the CDB are to be exported. To perform a full export, you must have the DATAPUMP_EXP_FULL_DATABASE role.
- SCHEMAS=hr, oe: All data and metadata of the schemas are to be exported. This is the default mode for Export. By default, if you do not have the DATAPUMP_EXP_FULL_DATABASE role, then only your own schema gets exported. If you have the DATAPUMP_EXP_FULL_DATABASE role, then you can specify a list of schemas.

- TABLES=hr.employees, oe.sales: Only the specified set of tables, partitions, and their dependent objects are unloaded.
- TABLESPACES=tbs_app, tbs2: Only the tables contained in a specified set of tablespaces are unloaded. If a table is unloaded, then its dependent objects are also unloaded. Both object metadata and data are unloaded.
- TRANSPORT_TABLESPACES=tbs_app, tbs2: Only object metadata contained in the tablespaces will be exported from the source database into the target database. The data is stored in data files. Because the data files do not get transported with the dump file, they should be copied to the target database before starting the import.
- TRANSPORTABLE=ALWAYS and FULL=YES: Both modes used together export all objects and data necessary to create a complete copy of the database. To import the full transportable database, use the TRANSPORT_DATAFILES='datafile1','datafile2' parameter to tell import that it is a transportable-mode import and from which data files to get the actual data.

Data Pump Import Transformations

You can remap:

- Data files by using `REMAP_DATAFILE`
- Tablespaces by using `REMAP_TABLESPACE`
- Schemas by using `REMAP_SCHEMA`
- Tables by using `REMAP_TABLE`
- Data by using `REMAP_DATA`
- Directory by using `REMAP_DIRECTORY`

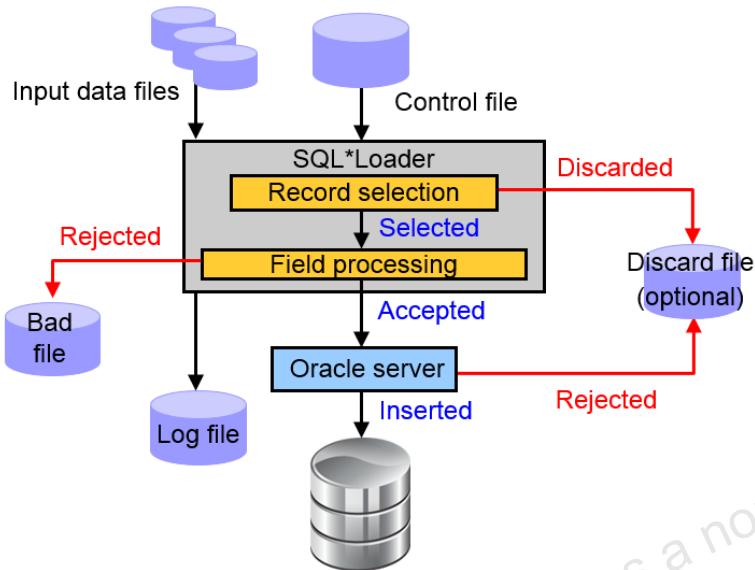


Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Because object metadata is stored as XML in the dump file set, it is easy to apply transformations when DDL is being formed during import. Data Pump Import supports several transformations:

- `REMAP_DATAFILE` is useful when moving databases across platforms that have different file-system semantics. It changes the name of the source data file to the target data file name in all SQL statements where the source data file is referenced: `CREATE TABLESPACE`, `CREATE LIBRARY`, and `CREATE DIRECTORY`.
- `REMAP_TABLESPACE` enables objects to be moved from one tablespace to another.
- `REMAP_SCHEMA` provides the capability to change object ownership.
- `REMAP_TABLE` provides the ability to rename entire tables.
- `REMAP_DATA` provides the ability to remap data as it is being inserted.
- `REMAP_DIRECTORY` provides the ability to remap directories when you move databases between platforms.

SQL Loader: Overview



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

SQL*Loader loads data from external files into the tables of an Oracle database. It has a powerful data parsing engine that puts little limitation on the format of the data in the data file.

SQL*Loader uses the following files:

Input data files: SQL*Loader reads data from one or more files (or operating system equivalents of files) that are specified in the control file. From SQL*Loader's perspective, the data in the data file is organized as records. A particular data file can be in fixed record format, variable record format, or stream record format. The record format can be specified in the control file with the `INFILE` parameter. If no record format is specified, the default is stream record format.

A control file: The control file is a text file that is written in a language that SQL*Loader understands. The control file indicates to SQL*Loader where to find the data, how to parse and interpret the data, where to insert the data, and so on. Although not precisely defined, a control file can be said to have three sections.

- The first section contains such session-wide information as the following:
 - Global options, such as the input data file name and records to be skipped
 - `INFILE` clauses to specify where the input data is located
 - Data to be loaded
- The second section consists of one or more `INTO TABLE` blocks. Each of these blocks contains information about the table (such as the table name and the columns of the table) into which the data is to be loaded.
- The third section is optional and, if present, contains input data.

A log file: When SQL*Loader begins execution, it creates a log file. If it cannot create a log file, execution terminates. The log file contains a detailed summary of the load, including a description of any errors that occurred during the load.

A discard file: This file is created only when it is needed and only if you have specified that a discard file should be enabled. The discard file contains records that are filtered out of the load because they do not match any record-selection criteria specified in the control file.

A bad file: The bad file contains records that are rejected, either by SQL*Loader or by the Oracle database. Data file records are rejected by SQL*Loader when the input format is invalid. After a data file record is accepted for processing by SQL*Loader, it is sent to the Oracle database for insertion into a table as a row. If the Oracle database determines that the row is valid, the row is inserted into the table. If the row is determined to be invalid, the record is rejected, and SQL*Loader puts it in the bad file.

For more information about SQL*Loader, see the Oracle Database Utilities guide.

Comparing Loading Methods

Conventional Load	Direct Path Load
Uses COMMIT	Uses data saves (faster operation)
Always generates redo entries	Generates redo only under specific conditions
Enforces all constraints	Enforces only PRIMARY KEY, UNIQUE, and NOT NULL constraints
Fires INSERT triggers	Does not fire INSERT triggers
Can load into clustered tables	Does not load into clusters
Allows other users to modify tables during load operation	Prevents other users from making changes to tables during load operation
Maintains index entries on each insert	Merges new index entries at the end of the load

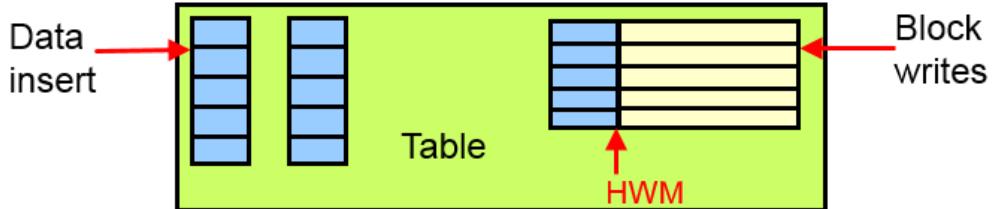


Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

A conventional path load executes SQL `INSERT` statements to populate tables in an Oracle database. Conventional path loads use SQL processing and a database `COMMIT` operation for saving data. The insertion of an array of records is followed by a `COMMIT` operation. Each data load may involve several transactions.

Direct path loads use data saves to format data blocks and write them directly to the data files bypassing the cache layer. This is why the direct path loads are faster than the conventional ones.

Data Save Feature



ORACLE®

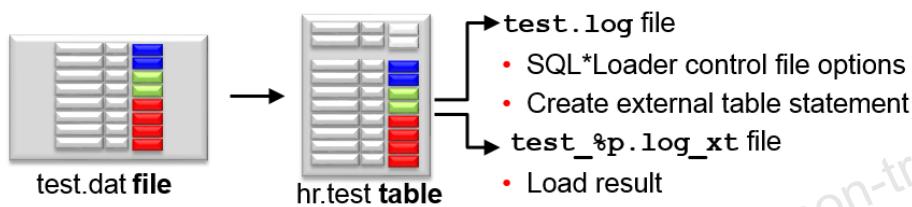
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The following features differentiate a data save from COMMIT:

- During a data save, only full database blocks are written to the database.
- The blocks are written after the high-water mark (HWM) of the table, as shown in the diagram in the slide.
- After a data save, the HWM is moved.
- Internal resources are not released after a data save.
- A data save does not end the transaction.
- If the ROWS parameter is specified, then SQL Loader issues a data save after that many rows are loaded.
- Indexes are not updated at each data save. At the beginning of a direct path load, all indexes on the table are marked unusable. The indexes are updated at the end of the load. If the load is aborted after the indexes were marked unusable, then they will remain unusable.
- The control file option that lets the direct path API handle check constraints is EVALUATE CHECK CONSTRAINTS.

Express Mode

- Specify a table name to initiate an Express Mode load.
- Table columns must be scalar data types (character, number, or datetime).
- A data file can contain only delimited character data.
- SQL*Loader uses table column definitions to determine input data types.
- There is no need to create a control file.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If you activate SQL*Loader Express Mode, specifying only the username and the TABLE parameter, it uses default settings for several other parameters. You can override most of the defaults by specifying additional parameters on the command line.

- The TERMINATED_BY parameter, which specifies a field terminator
- The ENCLOSED_BY parameter, which specifies a field enclosure character
- The OPTIONAL_ENCLOSURE_BY parameter, which specifies an optional field enclosure character

The three delimiters can be a multicharacter string.

SQL*Loader Express Mode generates two files. The names of the log files come from the name of the table (by default).

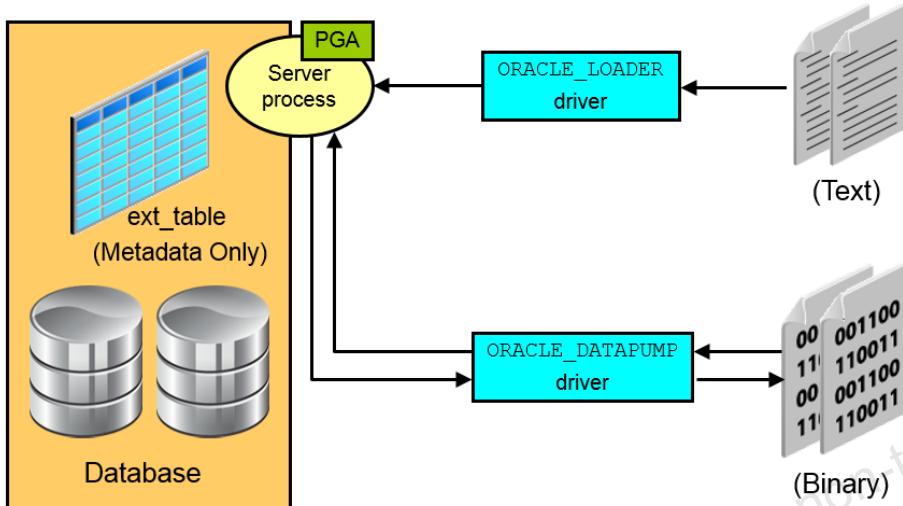
- A log file includes:
 - The control file output
 - A SQL script for creating the external table and performing the load by using a SQL INSERT /*+ APPEND */ AS SELECT statement

Neither the control file nor the SQL script are used by SQL*Loader Express Mode. They are made available to you in case you want to use them as a starting point to perform operations using regular SQL*Loader or stand-alone external tables.

You can specify that direct path load be used instead of external tables with the DIRECT=YES parameter. You can also specify that conventional path be used instead of external tables with DIRECT=NO.

- A log file similar to a SQL*Loader log file that describes the result of the operation. "%p" represents the process ID of the SQL*Loader process.

External Tables



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

External tables access data in external sources as if it were in a table in the database. You can connect to the database and create metadata for the external table by using DDL. The DDL for an external table consists of two parts:

- One part that describes the Oracle Database column types
- Another part that describes the mapping of the external data to the Oracle Database data columns

An external table does not describe any data that is stored in the database, nor does it describe how data is stored in the external source. Instead, it describes how the external table layer must present the data to the server. It is the responsibility of the access driver and the external table layer to do the necessary transformations required on the data in the external file so that it matches the external table definition. External tables are read-only; therefore, no DML operations are possible.

There are two access drivers used with external tables.

- The ORACLE_LOADER access driver can be used only to read table data from an external table and load it into the database. It uses text files as the data source.
- The ORACLE_DATAPUMP access driver can both load table data from an external file into the database and also unload data from the database into an external file. It uses binary files as the external files.

As of Oracle Database 12c Release 2 (12.2.0.1), you can partition data contained in external tables, which allows you to take advantage of the same performance improvements provided when you partition tables stored in a database.

External Table Benefits

- Data can be used directly from the external file or loaded into another database.
- External data can be queried and joined directly in parallel with tables residing in the database, without requiring it to be loaded first.
- The results of a complex query can be unloaded to an external file.
- You can combine generated files from different sources for loading purposes.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The data files created for the external table can be moved and used as the data files for another external table in the same database or a different database.

External data can be queried and joined directly in parallel to tables residing in the database, without requiring the data to be loaded first. You can choose to have your applications directly access external tables with the `SELECT` command, or you can choose to have data loaded first into a target database.

The results of a complex query can be unloaded to an external file by using the `ORACLE_DATAPUMP` access driver.

Data files that are populated by different external tables can all be specified in the `LOCATION` clause of another external table. This provides an easy way of aggregating data from multiple sources. The only restriction is that the metadata for all the external tables must be exactly the same.

Migrating to Oracle Database Cloud Service: Considerations

Some of the characteristics and factors to consider when choosing a migration method are:

- On-premises database version
- Oracle Database Cloud database version
- On-premises host operating system and version
- On-premises database character set
- Quantity of data, including indexes
- Data types used in the on-premises database
- Storage for data staging
- Acceptable length of system outage
- Network bandwidth



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Attend the *Migrating Your Oracle Database to the Oracle Cloud* course to learn more about migrating to Oracle Database Cloud Service.

Migrating to Oracle Database Cloud Service: Information Gathering

To determine which migration methods are applicable to your migration scenario, gather the following information:

- Database version of your on-premises database
- For on-premises Oracle Database 12c databases, the architecture of the database (multitenant or non-CDB)
- Endian format (byte ordering) of your on-premises database's host platform
- Database character set of your on-premises database and your Database Cloud Service database
- Database version of your Database Cloud Service database



See [Choosing a Migration Method](#) in *Using Oracle Database Cloud Service* for additional information.



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Applicable Migration Methods



Method	On-premises 11g database to Cloud 11g database	On-premises 11g database to Cloud 12c PDB	On-premises 12c non-CDB to Cloud 12c PDB	On-premises 12c PDB to Cloud 12c PDB
Data Pump Conventional Export/Import	Y	Y	Y	Y
Data Pump Transportable Tablespace	Y	Y	Y	Y
Data Pump Full Transportable	N	Y	Y	Y
RMAN Transportable Tablespace with Data Pump	Y	Y	Y	Y
RMAN CONVERT Transportable Tablespace with Data Pump	Y	Y	Y	Y
RMAN Cross-Platform Transportable Tablespace Backup Sets	N	N	Y	Y
RMAN Cross-Platform Transportable PDB	N	N	N	Y



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The methods shown in the table are applicable when migrating an Oracle Database 11g or 12c on-premises databases to an Oracle Database 11g or 12c database in the cloud.

Applicable Migration Methods



Method	On-premises 11g database to Cloud 11g database	On-premises 11g database to Cloud 12c PDB	On-premises 12c non-CDB to Cloud 12c PDB	On-premises 12c PDB to Cloud 12c PDB
Unplugging/Plugging	N	N	Y	Y
Remote Cloning	N	N	Y	Y
SQL Developer and SQL*Loader to Migrate Selected Objects	N	N	Y	Y
SQL Developer and <code>INSERT</code> Statements to Migrate Selected Objects	N	N	Y	Y



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The methods shown in the table are applicable when migrating an Oracle Database 11g or 12c on-premises databases to an Oracle Database 11g or 12c database in the cloud.

Summary

In this lesson, you should have learned how to:

- Describe ways to move data
- Explain the general architecture of Oracle Data Pump
- Use Data Pump Export and Import to move data between Oracle databases
- Use SQL*Loader to load data from a non-Oracle database (or user files)
- Use external tables to move data via platform-independent files
- Describe methods that can be used to migrate databases to Oracle Database Cloud Service



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 15: Overview

- 15-1: Moving Data from One PDB to Another
- 15-2: Loading Data into a PDB from an External File



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Backup and Recovery Concepts

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Identify DBA responsibilities regarding database backup and recovery
- Identify the types of failure that can occur in an Oracle database
- Describe instance recovery
- Describe complete and incomplete recovery



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

DBA Responsibilities

- Protect the database from failure wherever possible.
- Increase the mean time between failures (MTBF).
- Protect critical components by using redundancy.
- Decrease the mean time to recover (MTTR).
- Minimize the loss of data.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The database administrator (DBA) is typically responsible for ensuring that the database is open and available when users need it. To achieve that goal, the DBA (working with the system administrator):

- Anticipates and works to protect the database from failure wherever possible and to avoid common causes of failure
- Works to increase the mean time between failures (MTBF) that negatively affect availability
- Ensures that hardware is as reliable as possible, that critical components are protected by redundancy, and that operating system maintenance is performed in a timely manner. Oracle Database provides advanced configuration options to increase MTBF, including:
 - Real Application Clusters
 - Oracle Data Guard
- Decreases the mean time to recover (MTTR) by practicing recovery procedures in advance and configuring backups so that they are readily available when needed
- Minimizes the loss of data. DBAs who follow accepted best practices can configure their databases so that no committed transaction is ever lost. Entities that assist in guaranteeing this include:
 - Archive log files (discussed later in this lesson)
 - Flashback technology
 - Standby databases and Oracle Data Guard

Categories of Failure

Failures can generally be divided into the following categories:

- Statement failure
- User process failure
- Network failure
- User error
- Instance failure
- Media failure



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Failures can generally be divided into the following categories:

- **Statement failure:** A single database operation (select, insert, update, or delete) fails
- **User process failure:** A single database session fails
- **Network failure:** Connectivity to the database is lost
- **User error:** A user successfully completes an operation, but the operation (dropping a table or entering bad data) is incorrect
- **Instance failure:** The database instance shuts down unexpectedly
- **Media failure:** A loss of any file that is needed for database operation (that is, the files have been deleted or the disk has failed)

Statement Failure

Typical Problems	Possible Solutions
Attempts to enter invalid data into a table	Work with users to validate and correct data.
Attempts to perform operations with insufficient privileges	Provide the appropriate object or system privileges.
Attempts to allocate space that fails	Enable resumable space allocation. Increase owner quota. Add space to the tablespace.
Logic errors in applications	Work with developers to correct program errors.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When a single database operation fails, DBA involvement may be necessary to correct errors with user privileges or database space allocation. DBAs may also need to assist in troubleshooting, even for problems that are not directly in their task area. This can vary greatly from one organization to another.

For example, in organizations that use off-the-shelf applications (that is, organizations that have no software developers), the DBA is the only point of contact and must examine logic errors in applications.

To understand logic errors in applications, you should work with developers to understand the scope of the problem. Oracle Database tools may provide assistance by helping to examine audit trails or previous transactions.

Note: In many cases, statement failures are by design and desired. For example, security policies and quota rules are often decided upon in advance. If a user gets an error while trying to exceed his or her limits, it may be better for the operation to fail and no resolution may be necessary.

User Process Failure

Typical Problems	Possible Solutions
A user performs an abnormal disconnect.	A DBA's action is not usually needed to resolve user process failures.
A user's session is abnormally terminated.	Instance background processes roll back uncommitted changes and release locks.
A user experiences a program error that terminates the session.	The DBA should watch for trends.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

User processes that abnormally disconnect from the instance may have uncommitted work in progress that needs to be rolled back. The Process Monitor (PMON) background process periodically polls server processes to ensure that their sessions are still connected. If PMON finds a server process whose user is no longer connected, PMON recovers from any ongoing transactions; it also rolls back uncommitted changes and releases any locks that are held by the failed session.

A DBA's intervention should not be required to recover from user process failure, but the administrator must watch for trends. One or two users disconnecting abnormally is not a cause for concern. A small percentage of user process failures may occur from time to time.

But consistent and systemic failures indicate other problems. A large percentage of abnormal disconnects may indicate a need for user training (which includes teaching users to log out rather than just terminate their programs). It may also be indicative of network or application problems.

Network Failure

Typical Problems	Possible Solutions
Listener fails	Configure a backup listener and connect-time failover.
Network interface card (NIC) fails	Configure multiple network cards.
Network connection fails.	Configure a backup network connection.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The best solution to network failure is to provide redundant paths for network connections. Backup listeners, network connections, and network interface cards reduce the chance that network failures will affect system availability.

User Error

Typical Problems	Possible Solutions
User inadvertently deletes or modifies data	Roll back a transaction and dependent transactions or rewind the table
User drops a table	Recover the table from recycle bin Recover the table from a backup

Use Oracle LogMiner to query your online redo logs and archived redo logs through an Enterprise Manager or SQL interface.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Users may inadvertently delete or modify data. If they have not yet committed or exited their program, they can simply roll back.

You can use Oracle LogMiner to query your online redo logs and archived redo logs through an Enterprise Manager or SQL interface. Transaction data may persist in online redo logs longer than it persists in undo segments; if you have configured archiving of redo information, redo persists until you delete the archived files. Oracle LogMiner is discussed in *Oracle Database Utilities*.

Users who drop a table can recover it from the recycle bin by flashing back the table to before the drop.

If the recycle bin has already been purged, or if the user dropped the table with the PURGE option, the dropped table can still be recovered by using point-in-time recovery (PITR) if the database has been properly configured.

RMAN enables you to recover one or more tables or table partitions to a specified point in time without affecting the remaining database objects.

Note: Flashback technologies, PITR, and table recovery are discussed in the Oracle Database 18c: Backup and Recovery Workshop course and in *Oracle Database Backup and Recovery User's Guide*.

Instance Failure

Typical Causes	Possible Solutions
Power outage	Restart the instance by using the <code>STARTUP</code> command. Recovering from instance failure is automatic, including rolling forward changes in the redo logs and then rolling back any uncommitted transactions.
Hardware failure	Investigate the causes of failure by using the alert log, trace files, and Enterprise Manager.
Failure of one of the critical background processes	
Emergency shutdown procedures	



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Instance failure occurs when the database instance is shut down before synchronizing all database files. An instance failure can occur because of hardware or software failure or through the use of the emergency `SHUTDOWN ABORT` and `STARTUP FORCE` shutdown commands.

Administrator involvement in recovering from instance failure is rarely required if Oracle Restart is enabled and is monitoring your database. Oracle Restart attempts to restart your database instance as soon as it fails. If manual intervention is required, then there may be a more serious problem that prevents the instance from restarting, such as a memory CPU failure.

Media Failure

Typical Causes	Possible Solutions
Failure of a disk drive	Restore the affected file from backup.
Failure of a disk controller	Inform the database about a new file location (if necessary).
Deletion or corruption of a file needed for a database operation	Recover the file by applying redo information (if necessary).



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Corporation defines media failure as any failure that results in the loss or corruption of one or more database files (data, control, or redo log file).

Recovering from media failure requires that you restore and recover the missing files.

Understanding Instance Recovery

You can understand instance recovery by becoming familiar with these concepts and procedures:

- The checkpoint (CKPT) process
- Redo log files and the Log Writer (LGWR) process
- Automatic instance or crash recovery
- Phases of instance recovery
- Tuning instance recovery
- Using the MTTR Advisor



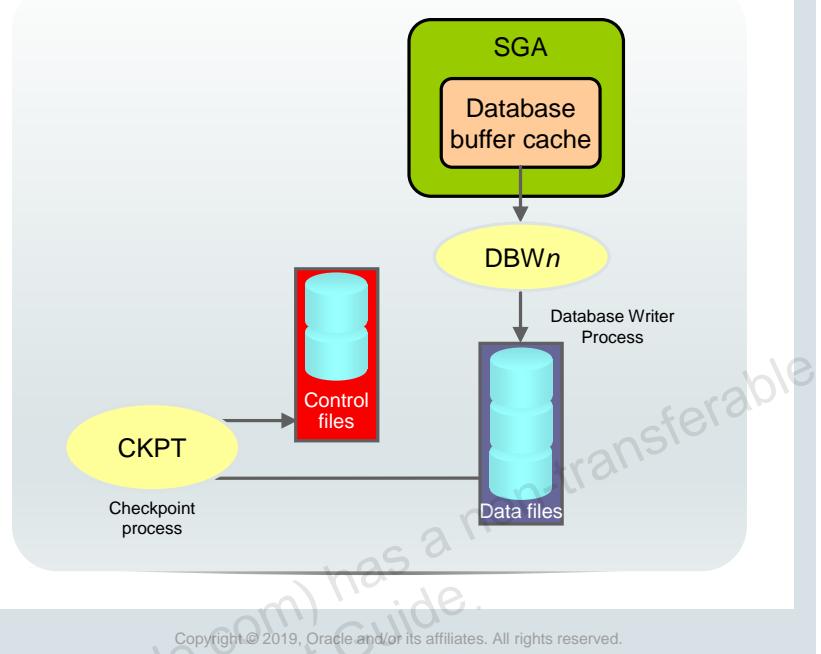
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Details on each of these topics follow on the next few pages.

The Checkpoint (CKPT) Process

CKPT is responsible for:

- Updating data file headers with checkpoint information
- Updating control files with checkpoint information
- Signaling DBW n at full checkpoints



To understand instance recovery, you need to understand the functioning of certain background processes.

Every three seconds (or more frequently), the CKPT process stores data in a control file to document the modified data blocks that DBW n has written from the SGA to disk. This is called an “incremental checkpoint.” The purpose of a checkpoint is to identify that place in the online redo log file where instance recovery is to begin (which is called the “checkpoint position”).

In the event of a log switch, the CKPT process also writes this checkpoint information to the headers of data files.

Checkpoints exist for the following reasons:

- To ensure that modified data blocks in memory are written to the disk regularly so that data is not lost in case of a system or database failure
- To reduce the time required for instance recovery (only the online redo log file entries following the last checkpoint need to be processed for recovery)
- To ensure that all committed data has been written to data files during shutdown

The checkpoint information written by the CKPT process includes checkpoint position, system change number (SCN), location in the online redo log file to begin recovery, information about logs, and so on.

Note: The CKPT process does not write data blocks to the disk or redo blocks to the online redo log files.

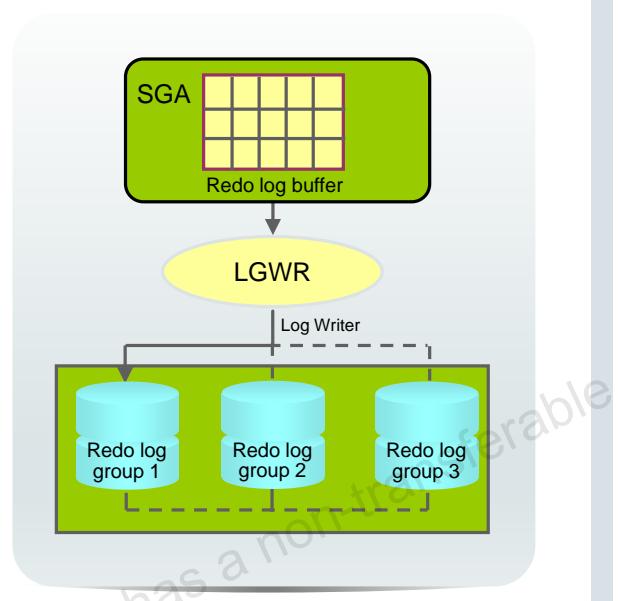
Redo Log Files and the Log Writer (LGWR) Process

Redo log files:

- Record changes to the database
- Should be multiplexed to protect against loss

Log Writer (LGWR) writes:

- At commit
- When one-third full
- Every three seconds
- Before DBWn writes
- Before clean shutdowns



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Redo log files record changes to the database as a result of transactions and internal Oracle server actions. A transaction is a logical unit of work consisting of one or more SQL statements run by a user. Redo log files protect the database from loss of integrity because of system failures caused by power outages, disk failures, and so on. Redo log files should be multiplexed to ensure that the information stored in them is not lost in the event of a disk failure.

The redo log consists of groups of redo log files. A group consists of a redo log file and its multiplexed copies. Each identical copy is said to be a member of that group, and each group is identified by a number. The Log Writer (LGWR) process writes redo records from the redo log buffer to all members of a redo log group until the files are filled or a log switch operation is requested.

It then switches and writes to the files in the next group. Redo log groups are used in a circular fashion.

Best practice tip: If possible, multiplexed redo log files should reside on different disks.

Automatic Instance Recovery or Crash Recovery

Automatic instance or crash recovery:

- Is caused by attempts to open a database whose files are not synchronized on shutdown
- Uses information stored in redo log groups to synchronize files
- Involves two distinct operations:
 - Rolling forward: Redo log changes (both committed and uncommitted) are applied to data files.
 - Rolling back: Changes that are made but not committed are returned to their original state.

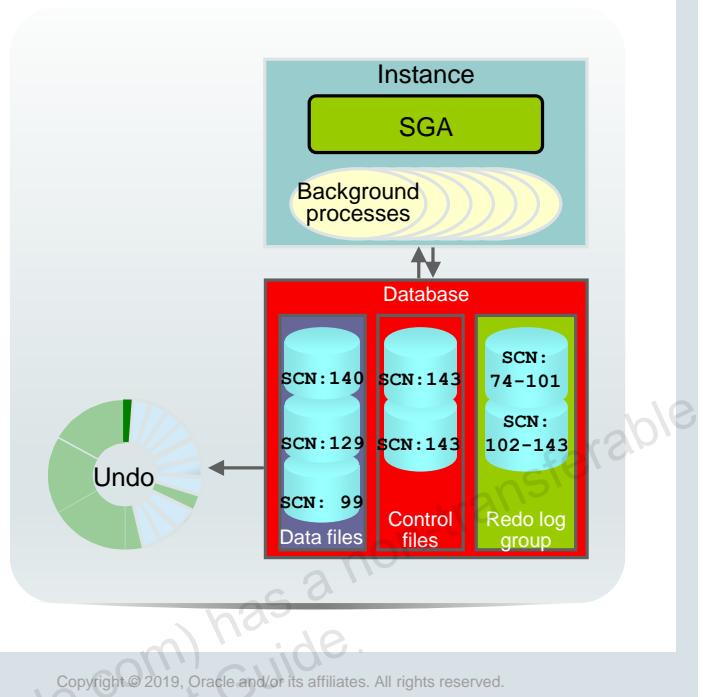


Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Oracle database server automatically recovers from instance failure. All that needs to happen is for the instance to be started normally. If Oracle Restart is enabled and configured to monitor this database, then this happens automatically. The instance mounts the control files and then attempts to open the data files. When it discovers that the data files have not been synchronized during shutdown, the instance uses information contained in the redo log groups to roll the data files forward to the time of shutdown. Then the database is opened, and any uncommitted transactions are rolled back.

Phases of Instance Recovery

1. Instance startup (data files are out of sync)
2. Roll forward (redo)
3. Committed and uncommitted data in files
4. Database opened
5. Roll back (undo)
6. Committed data in files



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

For an instance to open a data file, the system change number (SCN) contained in the data file's header must match the current SCN that is stored in the database's control files.

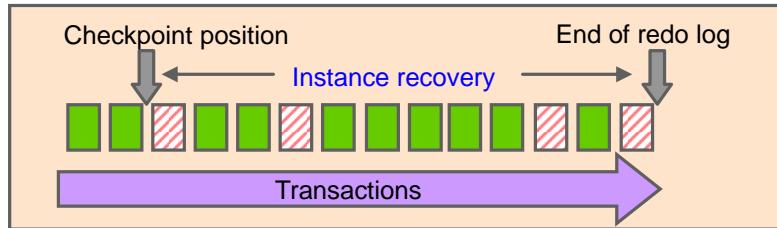
If the numbers do not match, the instance applies redo data from the online redo logs, sequentially "redoing" transactions until the data files are up-to-date. After all data files have been synchronized with the control files, the database is opened and users can log in.

When redo logs are applied, *all* transactions are applied to bring the database up to the state as of the time of failure. This usually includes transactions that are in progress but have not yet been committed. After the database has been opened, those uncommitted transactions are rolled back.

At the end of the rollback phase of instance recovery, the data files contain only committed data.

Tuning Instance Recovery

- During instance recovery, the transactions between the checkpoint position and the end of the redo log must be applied to data files.
- You tune instance recovery by controlling the difference between the checkpoint position and the end of the redo log.



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Transaction information is recorded in the redo log groups before the instance returns `commit` complete for a transaction. The information in the redo log groups guarantees that the transaction can be recovered in case of a failure. The transaction information must also be written to the data file. The data file write usually happens at some time after the information is recorded in redo log groups because the data file write process is much slower than the redo writes. Random writes for data files are slower than serial writes for redo log files.

Every three seconds, the checkpoint process records information in the control file about the checkpoint position in the redo log. Therefore, the Oracle Database server knows that all redo log entries recorded before this point are not necessary for database recovery. In the graphic in the slide, the striped blocks have not yet been written to the disk.

The time required for instance recovery is the time required to bring data files from their last checkpoint to the latest SCN recorded in the control file. The administrator controls that time by setting an MTTR target (in seconds) and through the sizing of redo log groups. For example, for two redo groups, the distance between the checkpoint position and the end of the redo log group cannot be more than 90% of the smallest redo log group.

Using the MTTR Advisor

- Specify the desired time in seconds or minutes.
- The default value is 0 (disabled).
- The maximum value is 3,600 seconds (one hour).

The screenshot shows the 'Recovery Settings' page in Oracle Enterprise Manager Cloud Control. At the top right, it says 'Logged in as DBA1'. Below that are three buttons: 'Show SQL', 'Revert', and 'Apply'. The main section is titled 'Instance Recovery'. It contains a detailed description of the fast-start checkpointing feature and its relationship to the FAST_START_MTTR_TARGET parameter. A note states that setting the value to 0 will disable this functionality. Below the text, there is a form field labeled 'Desired Mean Time To Recover' with a value of '0' and a dropdown menu set to 'Minutes'.

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The `FAST_START_MTTR_TARGET` initialization parameter simplifies the configuration of recovery time from instance or system failure. The MTTR Advisor converts the `FAST_START_MTTR_TARGET` value into several parameters to enable instance recovery in the desired time (or as close to it as possible). Note that explicitly setting the `FAST_START_MTTR_TARGET` parameter to 0 disables the MTTR Advisor.

The `FAST_START_MTTR_TARGET` parameter must be set to a value that supports the service level agreement for your system. A small value for the MTTR target increases I/O overhead because of additional data file writes (affecting the performance). However, if you set the MTTR target too large, the instance takes longer to recover after a crash.

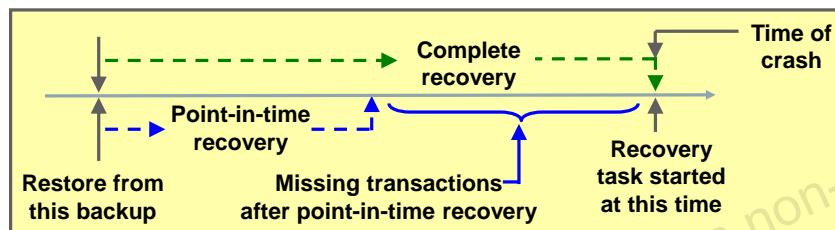
For assistance in setting the MTTR target by using Enterprise Manager Cloud Control, navigate as follows:

- Performance > Advisors Home > MTTR Advisor
- Availability > Backup & Recovery > Recovery Settings

Comparing Complete and Incomplete Recovery

Recovery can have two kinds of scope:

- Complete recovery: Brings the database or tablespace up to the present, including all committed data changes made to the point in time when the recovery was requested
- Incomplete or point-in-time recovery (PITR): Brings the database or tablespace up to a specified point in time in the past, before the recovery operation was requested

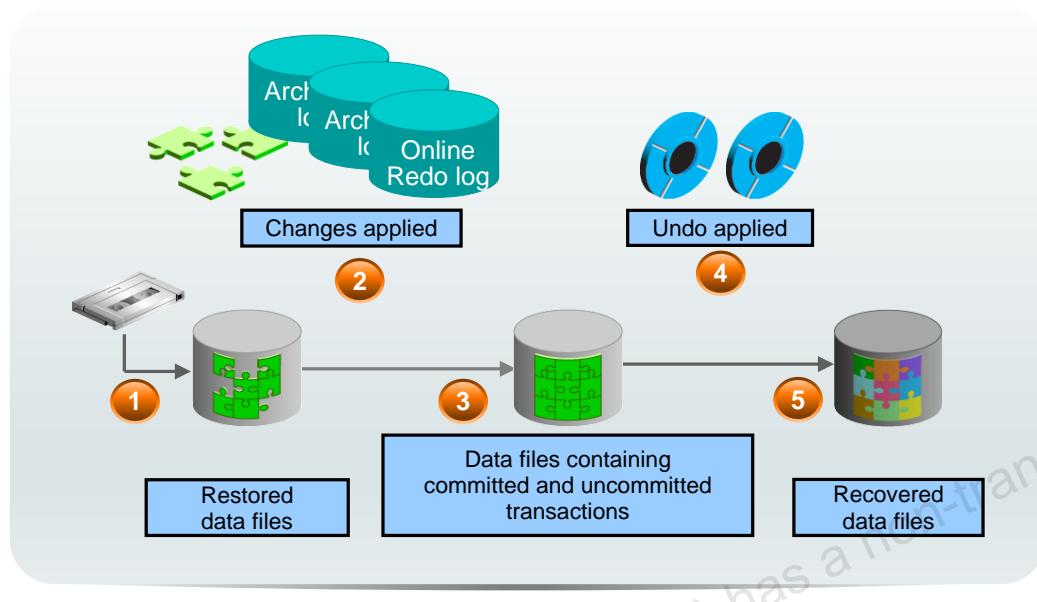


Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When you perform complete recovery, you bring the database to the state where it is fully up-to-date, including all committed data modifications to the present time.

Incomplete recovery, however, brings the database or tablespace to some point of time in the past. This is also known as “point-in-time recovery (PITR).” It means there are missing transactions; any data modifications done between the recovery destination time and the present are lost. In many cases, this is the desirable goal because there may have been some changes made to the database that need to be undone. Recovering to a point in the past is a way to remove the unwanted changes.

The Complete Recovery Process



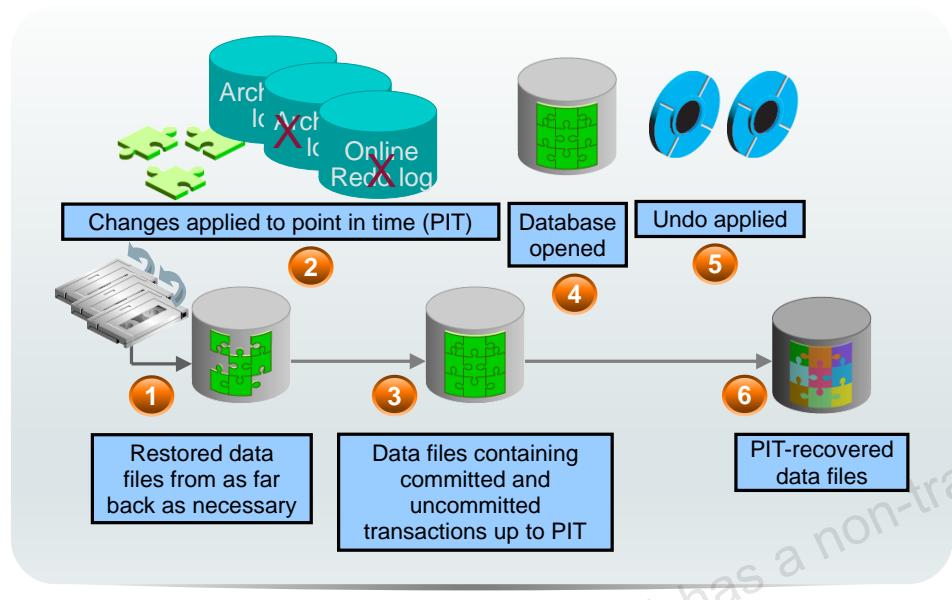
ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The following steps describe what takes place during complete recovery:

1. Damaged or missing files are restored from a backup.
2. Changes from incremental backups, archived redo log files, and online redo log files are applied as necessary. The redo log changes are applied to the data files until the current online log is reached and the most recent transactions have been re-entered. Undo blocks are generated during this entire process. This is referred to as rolling forward or cache recovery.
3. The restored data files may now contain committed and uncommitted changes.
4. The undo blocks are used to roll back any uncommitted changes. This is sometimes referred to as transaction recovery.
5. The data files are now in a recovered state and are consistent with the other data files in the database.

The Point-in-Time Recovery Process



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Incomplete recovery, or database point-in-time recovery (DBPITR), uses a backup to produce a noncurrent version of the database. That is, you do not apply all the redo records generated after the most recent backup. Perform this type of recovery only when absolutely necessary. To perform point-in-time recovery, you need:

- A valid offline or online backup of all the data files made before the recovery point
- All archived logs from the time of the backup until the specified time of recovery

The steps to perform a point-in-time recovery are as follows:

1. **Restore the data files from backup:** The backup that is used must be from before your target recovery point. This entails either copying files using OS commands or using the RMAN RESTORE command.
2. **Use the RECOVER command:** Apply redo from the archived redo log files, including as many as necessary to reach the restore point destination.
3. **State of over-recovery:** Now the data files contain some committed and some uncommitted transactions because the redo can contain uncommitted data.
4. **Use the ALTER DATABASE OPEN command:** The database is opened before undo is applied. This is to provide higher availability.

5. **Apply undo data:** While the redo was being applied, redo supporting the undo data files was also applied. So the undo is available to be applied to the data files in order to undo any uncommitted transactions. That is done next.
6. **Process complete:** The data files are now recovered to the point in time that you chose.

Oracle Flashback Database is the most efficient alternative to DBPITR. Unlike the other flashback features, it operates at a physical level and reverts the current data files to their contents at a past time. The result is like the result of a DBPITR, including the `OPEN RESETLOGS`, but Flashback Database is typically faster because it does not require you to restore data files and requires only limited application of redo compared to media recovery.

Oracle Data Protection Solutions

Backup and Recovery Objective	Recovery Time Objective (RTO)	Oracle Solution
Physical data protection	Hours/Days	Recovery Manager Oracle Secure Backup
Logical data protection	Minutes/Hours	Flashback Technologies
Recovery analysis	Minimize time for problem identification and recovery planning	Data Recovery Advisor
Disaster Recovery Objective	Recovery Time Objective (RTO)	Oracle Solution
Physical data protection	Seconds/Minutes	Data Guard Active Data Guard



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

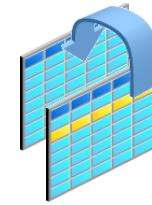
Oracle provides an appropriate data protection solution depending on your backup and recovery objective and RTO:

- Oracle Recovery Manager (RMAN) is the core Oracle Database software component that manages database backup, restore, and recovery processes.
- Oracle Secure Backup (OSB) is Oracle's enterprise-grade tape backup management solution for both database and file system data.
- Oracle Database Flashback technologies are a set of data recovery solutions that enable human errors to be reversed by selectively and efficiently undoing the effects of a mistake.
- The Data Recovery Advisor provides intelligent database problem identification and recovery capabilities.
- Data Guard and Active Data Guard enable physical standby databases to be open for read access while being kept synchronized with the production database through media recovery.

Flashback Technology

Use Flashback technology for:

- Viewing past states of data
- Winding data back and forth in time
- Assisting users in error analysis and recovery



For error analysis:

Oracle Flashback Query

Oracle Flashback Versions Query

Oracle Flashback Transaction Query

For error recovery:

Oracle Flashback Transaction Backout

Oracle Flashback Table

Oracle Flashback Drop

Oracle Flashback Database



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Oracle Database includes Oracle Flashback technology: a group of features that support viewing past states of data and winding data back and forth in time, without requiring the restoration of the database from backup. With this technology, you help users analyze and recover from errors. For users who have committed erroneous changes, use the following to analyze the errors:

- **Flashback Query:** View committed data as it existed at some point in the past. The SELECT command with the AS OF clause references a time in the past through a time stamp or system change number (SCN).
- **Flashback Version Query:** View committed historical data for a specific time interval. Use the VERSIONS BETWEEN clause of the SELECT command (for performance reasons with existing indexes).
- **Flashback Transaction Query:** View all database changes made at the transaction level.

Possible solutions to recover from user error:

- **Flashback Transaction Backout:** Rolls back a specific transaction and dependent transactions
- **Flashback Table:** Rewinds one or more tables to their contents at a previous time without affecting other database objects

Summary

In this lesson, you should have learned how to:

- Identify DBA responsibilities regarding database backup and recovery
- Identify the types of failure that can occur in an Oracle database
- Describe instance recovery
- Describe complete and incomplete recovery



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Backup and Recovery Configuration

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Configure the fast recovery area
- Multiplex the control file
- Multiplex redo log files
- Configure ARCHIVELOG mode



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Configuring for Recoverability

Configure your database for maximum recoverability by:

- Scheduling regular backups
- Multiplexing control files
- Multiplexing redo log groups
- Retaining archived copies of redo logs



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

To provide the best protection for your data, you must:

- **Schedule regular backups:** Most media failures require that you restore the lost or damaged file from backup.
- **Multiplex control files:** All control files associated with a database are identical. Recovering from the loss of a single control file is not difficult; recovering from the loss of *all* control files is much more challenging. Guard against losing all control files by having at least two copies.
- **Multiplex redo log groups:** To recover from instance or media failure, redo log information is used to roll data files forward to the last committed transaction. If your redo log groups rely on a single redo log file, the loss of that file means that data is likely to be lost. Ensure that there are at least two copies of each redo log group; if possible, each copy should be under different disk controllers.
- **Retain archived copies of redo logs:** If a file is lost and restored from backup, the instance must apply redo information to bring that file up to the latest SCN contained in the control file. With the default setting, the database can overwrite redo information after it has been written to the data files. Your database can be configured to retain redo information in archived copies of the redo logs. This is known as placing the database in ARCHIVELOG mode.

You can perform configuration tasks in Enterprise Manager Cloud Control or by using the SQL command line.

Configuring the Fast Recovery Area

- Fast recovery area:
 - Strongly recommended for simplified backup storage management
 - Storage space (separate from working database files)
 - Location specified by the `DB_RECOVERY_FILE_DEST` parameter
 - Size specified by the `DB_RECOVERY_FILE_DEST_SIZE` parameter
 - Large enough for backups, archived logs, flashback logs, multiplexed control files, and multiplexed redo logs
 - Automatically managed according to your retention policy
- Configuration of the fast recovery area includes specifying the location, size, and retention policy.

```
ALTER SYSTEM SET db_recovery_file_dest = directory | disk group  
ALTER SYSTEM SET db_recovery_file_destsize = integer [K | M | G]
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The fast recovery area is space that is set aside on disk to contain archived logs, backups, flashback logs, multiplexed control files, and multiplexed redo logs. A fast recovery area simplifies backup storage management and is strongly recommended. You should place the fast recovery area on storage space that is separate from the location of your database data files, primary online log files, and control file.

The amount of disk space to allocate for the fast recovery area depends on the size and activity levels of your database. As a general rule, the larger the fast recovery area, the more useful it is. Ideally, the fast recovery area should be large enough for copies of your data and control files and for flashback, online redo, and archived logs needed to recover the database with the backups kept based on the retention policy. In short, the fast recovery area should be at least twice the size of the database so that it can hold one backup and several archived logs.

Space management in the fast recovery area is governed by a backup retention policy. A retention policy determines when files are obsolete, which means that they are no longer needed to meet your data recovery objectives. The Oracle Database server automatically manages this storage by deleting files that are no longer needed.

Monitoring the Fast Recovery Area

- Monitor the fast recovery area to ensure that it does not reach its capacity.
- The instance will pause if there isn't enough space in the fast recovery area to create an archived log.
- Query the `V$RECOVERY_FILE_DEST` view to determine the current location, disk quota, space in use, space reclaimable by deleting files, and total number of files in the fast recovery area.
- Query the `V$RECOVERY_AREA_USAGE` view to determine the percentage of the total disk quota used by different types of files.
- You can also use GUI tools such as Enterprise Manager Cloud Control to monitor the space usage.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If you have configured your archived logs to be written to this location, it is important to monitor this space to ensure that it does not reach its capacity. If the instance is unable to create an archived log because of lack of space, it pauses until the administrator corrects the situation.

You can use the `V$RECOVERY_FILE_DEST` and `V$RECOVERY_AREA_USAGE` views to determine whether you have allocated enough space for your fast recovery area.

The retention time determines when files are obsolete (that is, when they are no longer needed to meet your data recovery objectives). The Oracle Database server automatically manages this storage, deleting files that are no longer needed. You can back up the recovery area so that Oracle Recovery Manager (RMAN) can fail over to other archived redo log destinations if the archived redo log in the fast recovery area is inaccessible or corrupted.

Periodically copying backups to tape frees space in the fast recovery area for other files, but retrieving files from tape causes longer database restoration and recovery times.

Multiplexing Control Files

To protect against database failure, your database should have multiple copies of the control file.

	ASM Storage	File System Storage
Best Practice	One copy on each disk group (such as +DATA and +FRA)	At least two copies, each on a separate disk (at least one on a separate disk controller)
Steps to create additional control files	No additional control file copies required	<ol style="list-style-type: none">1. Alter the SPFILE with the ALTER SYSTEM SET control_files command.2. Shut down the database.3. Copy the control file to a new location.4. Open the database and verify the addition of the new control file.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

A control file is a small binary file that describes the structure of the database. It must be available for writing by the Oracle server whenever the database is mounted or opened. Without this file, the database cannot be mounted, and recovery or re-creation of the control file is required. Your database should have a minimum of two control files on different storage devices to minimize the impact of a loss of one control file.

The loss of a single control file causes the instance to fail because all control files must be available at all times. However, recovery can be a simple matter of copying one of the other control files. The loss of all control files is slightly more difficult to recover from but is not usually catastrophic.

Adding a Control File

In a database using regular file system storage, adding a control file is a manual operation:

1. Alter the SPFILE with the following command specifying the appropriate location of your files:

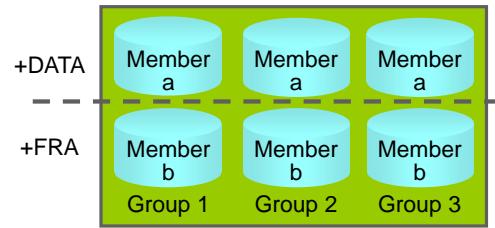
```
ALTER SYSTEM SET control_files =
  '/u01/app/oracle/oradata/orcl/control01.ctl' ,
  '/u02/app/oracle/oradata/orcl/control02.ctl' ,
  '/u03/app/oracle/oradata/orcl/control03.ctl' SCOPE=SPFILE;
```
2. Shut down the database instance.
3. Use an operating system command to copy an existing control file to the location you select for your new file.
4. Open the database.

If you are using ASM as your storage technique, then as long as you have two control files, one in each disk group (such as +DATA and +FRA), you should not require further multiplexing. In a database using Oracle Managed Files (OMF)—such as a database using ASM storage—all additional control files must be created as part of a recovery process using RMAN (or through Enterprise Manager).

Redo Log Files

Multiplex redo log groups to protect against media failure and loss of data. This increases database I/O. It is suggested that redo log groups have:

- At least two members (files) per group
- Each member:
 - On a separate disk or controller if using file system storage
 - In a separate disk group (such as +DATA and +FRA) if using ASM



Note: Multiplexing redo logs may impact overall database performance.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Redo log groups are made up of one or more redo log files. Each log file in a group is a duplicate of the others. Oracle Corporation recommends that redo log groups have at least two files per group. If you are using file system storage, then each member should be distributed on separate disks or controllers so that no single equipment failure impacts an entire log group. If you are using ASM storage, then each member should be in a separate disk group, such as +DATA and +FRA.

The loss of an entire current log group is one of the most serious media failures because it can result in loss of data. The loss of a single member of a multiple-member log group is trivial and does not affect database operation (other than causing an alert to be published in the alert log). Recovery from the loss of an entire log group requires advanced recovery techniques and is discussed in the course titled *Oracle Database 18c: Backup and Recovery Workshop*.

Remember that multiplexing redo logs may heavily influence database performance because a commit cannot complete until the transaction information has been written to the logs. You must place your redo log files on your fastest disks served by your fastest controllers. If possible, do not place any other database files on the same disks as your redo log files (unless you are using ASM). Because only one group is written to at a given time, there is no performance impact in having members from several groups on the same disk.

For information on file placement in a Database Cloud Service database deployment, see “Characteristics of a Newly Created Deployment” in *Administering Oracle Database Cloud Service*.

Multiplexing the Redo Log

Add a member to an existing log group:

- Navigate to the Redo Log Groups page in Enterprise Manager Database Express
- Use the `ALTER DATABASE` command

```
SQL> ALTER DATABASE
  2  ADD LOGFILE MEMBER '/u01/app/oracle/oradata/orcl/red01a.log'
  3  TO GROUP 1;
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can multiplex your redo log by adding a member to an existing log group. To add a member to a redo log group (with open database and no impact on user performance), perform the following steps in Enterprise Manager Database Express:

1. Select Storage > Redo Log Groups.
2. Select a group and click Add Member.
3. The Add Member page appears.
4. For File System storage, enter the file name and the file directory. Click OK.

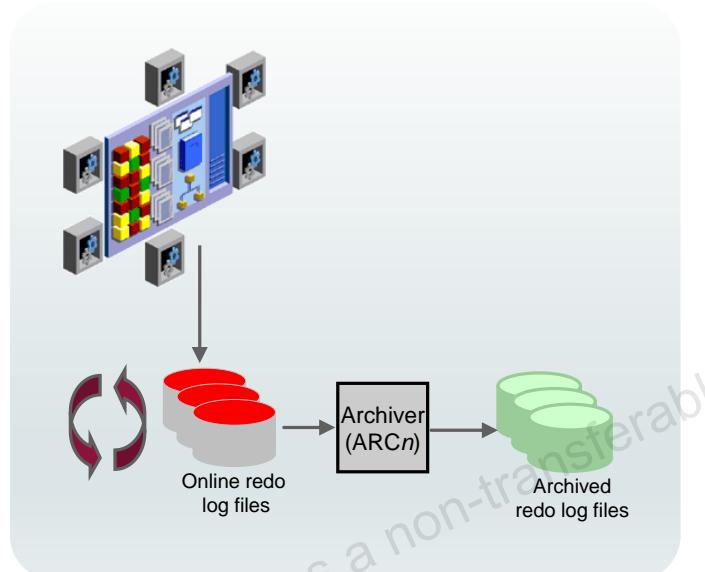
Repeat these steps for every existing group that you want to multiplex.

When you add the redo log member to a group, the member's status is marked as `INVALID` (as can be seen in the `V$LOGFILE` view). This is the expected state because the new member of the group has not yet been written to. When a log switch occurs and the group containing the new member becomes `CURRENT`, the member's status changes to null.

Creating Archived Redo Log Files

To preserve redo information, create archived copies of redo log files by performing the following steps:

1. Specify the archived redo log file-naming convention.
2. Specify one or more archived redo log file locations.
3. Place the database in ARCHIVELOG mode.



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Oracle Database server treats the online redo log groups as a circular buffer in which to store transaction information, filling one group and then moving on to the next. After all groups have been written to, the Oracle Database server begins overwriting information in the first log group.

To configure your database for maximum recoverability, you must instruct the Oracle Database server to make a copy of the online redo log group before allowing it to be overwritten. These copies are known as *archived redo log files*.

To facilitate the creation of archived redo log files:

1. Specify a naming convention for your archived redo log files.
2. Specify a destination or destinations for storing your archived redo log files.
3. Place the database in ARCHIVELOG mode.

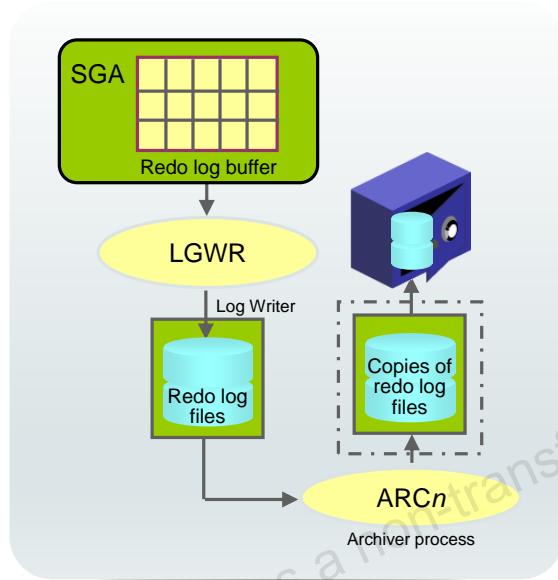
Note: Steps 1 and 2 are not necessary if you are using a fast recovery area.

The destination should exist before placing the database in ARCHIVELOG mode. When a directory is specified as a destination, there should be a slash at the end of the directory name.

Archiver (ARCn) Process

Archiver (ARCn):

- Automatically archives online redo log files when the database is in ARCHIVELOG mode
- Preserves a record of all changes made to the database



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

ARCn is a background process that is crucial to the recovery of a database after the loss of a disk. When an online redo log group gets filled, the Oracle Database server begins writing to the next online redo log group. The process of switching from one online redo log group to another is called a *log switch*. The ARCn process initiates archiving of the filled log group at every log switch. It automatically archives the online redo log group before the log group can be reused so that all the changes made to the database are preserved. This enables recovery of the database to the point of failure even if a disk drive is damaged.

One of the important decisions that a DBA must make is whether to configure the database to operate in ARCHIVELOG mode or in NOARCHIVELOG mode.

- In NOARCHIVELOG mode, the online redo log files are overwritten each time a log switch occurs.
- In ARCHIVELOG mode, inactive groups of filled online redo log files must be archived before they can be used again.

Note

- ARCHIVELOG mode is essential for most backup strategies.
- If the archived redo log file destination fills up or cannot be written to, the database will eventually come to a halt. Remove archived redo log files from the archived redo log file destination and the database will resume operations.

Archived Redo Log Files: Naming and Destinations

- Use the `LOG_ARCHIVE_DEST` initialization parameter to specify a single destination.
- Use the `LOG_ARCHIVE_DEST_n` initialization parameters to archive to two or more locations.
- If you are using file system storage, it is recommended that you add multiple locations across different disks.
- If the fast recovery area is enabled, `USE_DB_RECOVERY_FILE_DEST` is specified by default as an archived redo log file destination.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Each archived redo log file must have a unique name to avoid overwriting older log files. Specify the naming format as shown in the slide. To help create unique file names, Oracle Database allows several wildcard characters in the name format:

- `%s`: Includes the log sequence number as part of the file name
- `%t`: Includes the thread number as part of the file name
- `%r`: Includes the resetlogs ID to ensure that the archive log file name remains unique (even after certain advanced recovery techniques that reset log sequence numbers)
- `%d`: Includes the database ID as part of the file name

The format should include `%s`, `%t`, and `%r` as best practice (`%d` can also be included if multiple databases share the same archive log destination).

By default, if the fast recovery area is enabled, `USE_DB_RECOVERY_FILE_DEST` is specified as an archived redo log file destination. Archived redo log files can be written to as many as 10 different destinations. Destinations may be local (a directory) or remote (an Oracle Net alias for a standby database).

Configuring ARCHIVELOG Mode

- You can use SQL commands as follows:
 1. Shut down the database instance if it is open.
 2. Mount the database.
 3. Issue the ALTER DATABASE ARCHIVELOG command.
 4. Open the database.

```
SQL> shutdown immediate
SQL> startup mount
SQL> alter database archivelog;
SQL> alter database open;
```

- You can also use Enterprise Manager Cloud Control to place the database in ARCHIVELOG mode.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Placing the database in ARCHIVELOG mode prevents redo logs from being overwritten until they have been archived.

To issue the SQL command to put the database in ARCHIVELOG mode, the database must be in MOUNT mode. If the database is currently open, you must shut it down cleanly (not abort) and then mount it.

With the database in NOARCHIVELOG mode (the default), recovery is possible only until the time of the last backup. All transactions made after that backup are lost.

In ARCHIVELOG mode, recovery is possible until the time of the last commit. Most production databases are operated in ARCHIVELOG mode.

Note: Back up your database after switching to ARCHIVELOG mode because your database is recoverable only from the first backup taken in that mode.

Summary

In this lesson, you should have learned how to:

- Configure the Fast Recovery Area
- Multiplex the control file
- Multiplex redo log files
- Configure ARCHIVELOG mode



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 17: Overview

- 17-1: Verifying that the Control File is Multiplexed
- 17-2: Checking Storage Availability
- 17-3: Configuring the Size of the Fast Recovery Area
- 17-4: Verifying that the Redo Log File is Multiplexed
- 17-5: Verifying that ARCHIVELOG Mode is Configured



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Creating Database Backups

The ORACLE logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Create consistent database backups
- Back up your database without shutting it down
- Create incremental backups
- Modify the DBCS default backup configuration
- Create DBCS backups



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Understanding Types of Backups

You can understand different types of backups by becoming familiar with these concepts:

- Backup terminology
- Types of backups
- RMAN backup types



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Backup Terminology

- *Backup strategy* may include:
 - The entire database (whole)
 - A portion of the database (partial)
- *Backup type* may indicate inclusion of:
 - All data blocks within your chosen files (full)
 - Only information that has changed since a previous backup (incremental)
 - Cumulative (changes since last level 0)
 - Differential (changes since last incremental)
- *Backup mode* may be:
 - Offline (consistent, cold)
 - Online (inconsistent, hot)



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Whole database backup: Includes all data files and at least one control file (remember that all control files in a database are identical)

Partial database backup: May include zero or more tablespaces and zero or more data files; may or may not include a control file

Full backup: Makes a copy of each data block that contains data and that is within the files being backed up

Incremental backup: Makes a copy of all data blocks that have changed since a previous backup. Oracle Database supports two levels of incremental backup (0 and 1). A level 1 incremental backup can be one of two types: *cumulative* or *differential*. A cumulative backup backs up all changes since the last level 0 backup. A differential backup backs up all changes since the last incremental backup (which could be either a level 0 or level 1 backup). Change Tracking with RMAN supports incremental backups.

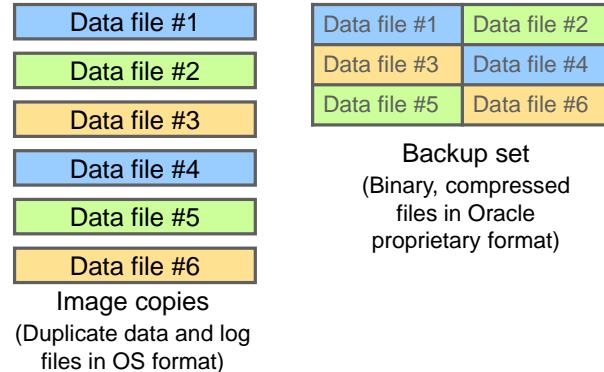
Offline backups (also known as “cold” or *consistent* backup): Are taken while the database is not open. They are consistent because, at the time of the backup, the system change number (SCN) in data file headers matches the SCN in the control files.

Online backups (also known as “hot” or *inconsistent* backup): Are taken while the database is open. They are inconsistent because, with the database open, there is no guarantee that the data files are synchronized with the control files.

Understanding Types of Backups

Backups may be stored as:

- Image copies
- Backup sets



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Image copies: Are duplicates of data or archived log files (similar to simply copying the files by using operating system commands)

Backup sets: Are collections of one or more binary files that contain one or more data files, control files, server parameter files, or archived log files. With backup sets, empty data blocks are not stored, thereby causing backup sets to use less space on the disk or tape. Backup sets can be compressed to further reduce the space requirements of the backup.

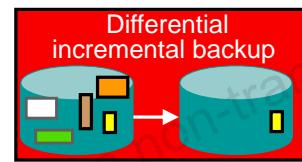
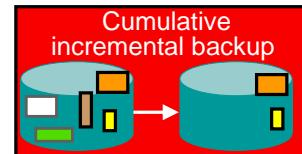
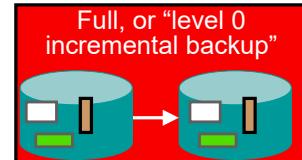
Image copies must be backed up to the disk. Backup sets can be sent to the disk or directly to the tape.

The advantage of creating a backup as an image copy is improved granularity of the restore operation. With an image copy, only the file or files need to be retrieved from your backup location. With backup sets, the entire backup set must be retrieved from your backup location before you extract the file or files that are needed.

The advantage of creating backups as backup sets is better space usage. In most databases, 20% or more of the data blocks are empty blocks. Image copies back up every data block, even if the data block is empty. Backup sets significantly reduce the space required by the backup. In most systems, the advantages of backup sets outweigh the advantages of image copies.

RMAN Backup Types

- A *full backup* contains all used data file blocks.
- A *level 0 incremental backup* is equivalent to a full backup that has been marked as level 0.
- A *cumulative level 1 incremental backup* contains only blocks modified since the last level 0 incremental backup.
- A *differential level 1 incremental backup* contains only blocks modified since the last incremental backup.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Full Backups

A full backup is different from a whole database backup. A full data file backup is a backup that includes every used data block in the file. RMAN copies all blocks into the backup set or image copy, skipping only those data file blocks that are not part of an existing segment. For a full image copy, the entire file contents are reproduced exactly. A full backup cannot be part of an incremental backup strategy; it cannot be the parent for a subsequent incremental backup.

Incremental Backups

An incremental backup is either a level 0 backup, which includes every block in the data files except blocks that have never been used, or a level 1 backup, which includes only those blocks that have been changed since a previous backup was taken. A level 0 incremental backup is physically identical to a full backup. The only difference is that the level 0 backup (as well as an image copy) can be used as the base for a level 1 backup, but a full backup can never be used as the base for a level 1 backup.

Incremental backups are specified using the `INCREMENTAL` keyword of the `BACKUP` command. You specify `INCREMENTAL LEVEL [0 | 1]`.

RMAN can create multilevel incremental backups as follows:

- **Differential:** Is the default type of incremental backup that backs up all blocks changed after the most recent incremental backup at either level 1 or level 0
- **Cumulative:** Backs up all blocks changed after the most recent backup at level 0

Examples

- To perform an incremental backup at level 0, use the following command:

```
RMAN> BACKUP INCREMENTAL LEVEL 0 DATABASE;
```

- To perform a differential incremental backup, use the following command:

```
RMAN> BACKUP INCREMENTAL LEVEL 1 DATABASE;
```

- To perform a cumulative incremental backup, use the following command:

```
RMAN> BACKUP INCREMENTAL LEVEL 1 CUMULATIVE DATABASE;
```

RMAN makes full backups by default if neither `FULL` nor `INCREMENTAL` is specified. Unused block compression causes never-written blocks to be skipped when backing up data files to backup sets, even for full backups.

A full backup has no effect on subsequent incremental backups and is not considered part of any incremental backup strategy, although a full image copy backup can be incrementally updated by applying incremental backups with the `RECOVER` command.

Note: It is possible to perform any type of backup (full or incremental) of a database that is in `NOARCHIVELOG` mode—if, of course, the database is not open. Note also that recovery is limited to the time of the last backup. The database can be recovered to the last committed transaction only when the database is in `ARCHIVELOG` mode.

Using Recovery Manager (RMAN)

- Provides a powerful control and scripting language
- Includes a published API that enables the interface with the most popular backup software
- Backs up data, control, the archived redo log, and server parameter files
- Backs up files to disk or tape
- Is integrated with Enterprise Manager Cloud Control
- Is used by Oracle Database Cloud Service, Oracle Database Backup Service, and other Oracle Cloud services for automated backups
- Can be used with Oracle Database Cloud Service if no backup configuration was selected when the database deployment was created



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

RMAN is the component of the Oracle Database server that is used to perform backup and recovery operations. It can be used to make consistent and inconsistent backups, perform incremental and full backups, and back up either the whole database or a portion of it.

RMAN uses its own powerful job control and scripting language, as well as a published API that interfaces RMAN with many popular backup software solutions.

RMAN can store backups on the disk for quick recovery or place them on the tape for long-term storage. For RMAN to store backups on the tape, you must either use Oracle Secure Backup or configure an interface to the tape device known as a media management library (MML).

By using RMAN commands or the Enterprise Manager Cloud Control interface, you can manage the persistent backup settings that are used for creating backups.

Enterprise Manager Cloud Control provides a graphical interface to the most commonly used RMAN functionality.

Advanced backup and recovery operations are accessible through RMAN's command-line client. For more information about advanced RMAN capabilities, see the course titled *Oracle Database 18c: Backup and Recovery Workshop* or consult the *Oracle Backup and Recovery User's Guide*.

Backing Up the Control File to a Trace File

- Control files can be backed up to a trace file, generating a SQL command to re-create the control file.
- Control file trace backups may be used to recover from the loss of all control files.

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In Enterprise Manager Database Express, select Storage > Control Files to manage your database's control files. Control files have an additional backup option; they may be backed up to a trace file. A control file trace backup contains the SQL statement that is required to re-create the control files in the event that all control files are lost.

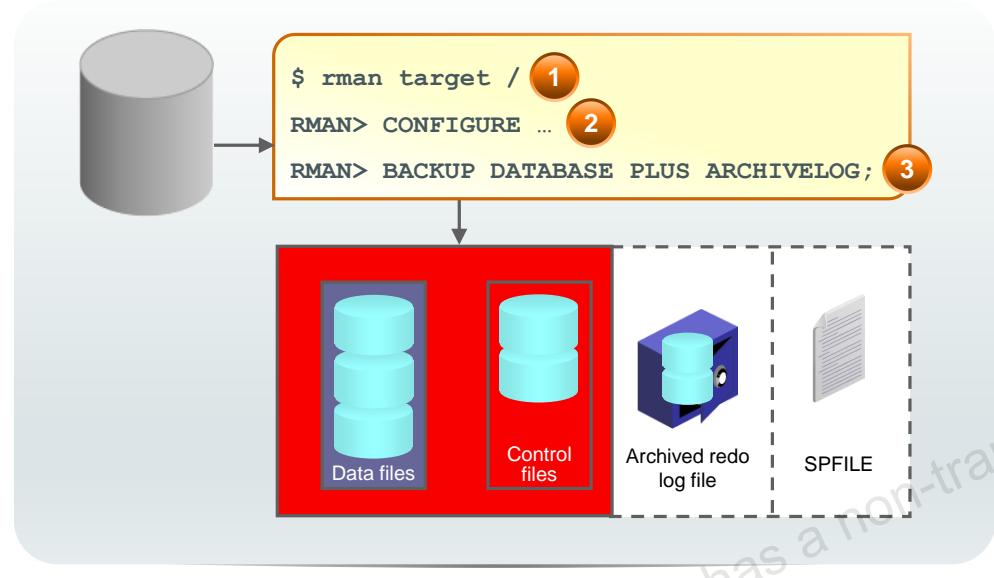
Although it is very unlikely that a properly configured database (with multiple copies of the control file placed on separate disks and separate controllers) would lose all control files at the same time, it is possible. Therefore, you should back up the control file to a trace file after each change to the physical structure of the database (adding tablespaces or data files or adding additional redo log groups).

Trace copies of the control file can be created by using Enterprise Manager Database Express, Enterprise Manager Cloud Control, or the following SQL command:

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE
```

The trace backup is created in the location specified by the `DIAGNOSTIC_DEST` initialization parameter. For example, in this course, the trace file for the `orcl` database is found in the `/u01/app/oracle/diag/rdbms/orcl/orcl/trace` directory and will have a file name such as `orcl_ora_924.trc`.

Using RMAN Commands to Create Backups



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

1. In a terminal session, start RMAN and connect to the target database.
2. Execute configuration commands:
 - CONFIGURE DEFAULT DEVICE TYPE TO disk;
 - CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COPY;
3. A whole database backup is a copy of all data files and the control file. You can optionally include the server parameter file (SPFILE) and archived redo log files. Using RMAN to make an image copy of all the database files simply requires mounting or opening the database, starting RMAN, and entering the BACKUP command shown in the slide.

Optionally, you can supply the `DELETE INPUT` option when backing up archive log files.

That causes RMAN to remove the archive log files after backing them up. This is useful especially if you are not using a fast recovery area, which would perform space management for you, deleting files when space pressure grows. In that case, the command in the slide would look like the following:

```
RMAN> BACKUP DATABASE PLUS ARCHIVELOG DELETE INPUT;
```

You can also create a backup (either a backup set or image copies) of previous image copies of all data files and control files in the database by using the following command:

```
RMAN> BACKUP COPY OF DATABASE;
```

Backing Up Databases on DBCS



- Database Cloud Service provides a backup feature that backs up:
 - The database
 - Database configuration files
 - Grid Infrastructure configuration files (on deployments hosting an Oracle RAC database)
 - System and cloud tooling files
- The backup feature relies on the following, which are installed in the database deployment:
 - System utilities
 - Oracle Database utilities
 - Oracle Database Backup Cloud Service

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

By backing up your Database Cloud Service database deployments, you can protect the software, configuration, and database against loss if a failure occurs. You can restore the deployment's software, configuration, and database to their state at the time of the backup.

To provide this backup feature, Database as a Service relies on system utilities, Oracle Database utilities, and Oracle Database Backup Cloud Service, all of which are installed in the database deployment.

Backup Destination Choices



Backup Destination	Description	Retention
Both Cloud Storage and Local Storage	Backups are created automatically and stored both on local compute node storage and on an Oracle Storage Cloud Service container.	30 days 7 most recent days' backups available on local storage
Cloud Storage Only	Backups are created automatically and stored only on an Oracle Storage Cloud Service container.	30 days
None	No backups are created.	



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When you create a database deployment, you choose one of the following backup destinations:

- Both Cloud Storage and Local Storage: Backups are configured to be created automatically, and stored both on local compute node storage and on an Oracle Storage Cloud Service container. The container must have been created before creating the Database as a Service database deployment.
- Cloud Storage Only: Backups are configured to be created automatically and stored only on an Oracle Storage Cloud Service container.
- None: No backups are created.

Backup Configuration



- Full (level 0) backup of the database followed by rolling incremental (level 1) backups on a seven-day cycle
- Full backup of selected database configuration files
- Full backup of selected system files
- Automatic backups daily at a time between 11 PM (23:00) and 3 AM (03:00), with the specific time set during database deployment creation
- Encryption:
 - Both Cloud Storage and Local Storage: All backups to cloud storage are encrypted; backups of Enterprise Edition databases to local storage are encrypted; backups of Standard Edition databases to local storage are not encrypted.
 - Cloud Storage Only: All backups to cloud storage are encrypted.

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The backup configuration created when you choose a destination other than None follows a set of Oracle best-practice guidelines, as listed in the slide.

Creating an On-Demand Backup



- Click “Backup Now” on the Oracle Database Cloud Service Instance Administration page.
- Command-line interfaces for backups:
 - For single-instance databases, use the `bkup_api` utility.
 - For RAC databases, use the `raccli` utility.

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can create an on-demand backup of a Database as a Service database deployment from the Oracle Database Cloud Service Instance Administration page.

You can also create on-demand backups by using command-line utilities. Use the `bkup_api` utility to create an on-demand backup on database deployments hosting single-instance databases. Use the `raccli` utility to create an on-demand backup on database deployments hosting Oracle RAC databases.

See “Creating an On-Demand Backup” in *Administering Oracle Database Cloud Service* for details.

Customizing the Backup Configuration



Customization	Description	Utility to Use or File to Edit
Backup Configuration Settings	Persistent configuration settings	Recovery Manager (RMAN)
System Files	System files and directories	/home/oracle/bkup/oscfg.spec file
Database Configuration Files	Wallet, initialization parameter, network configuration files	/home/oracle/bkup/dbcfg.spec file
Retention Period	Backup retention period (days)	bkup_api utility
Cycle Period	Backup cycle period (days)	bkup_api utility
Frequency	Time that bkup_api is run	/etc/crontab file



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can customize many of the characteristics of the backup configuration.

To change how the Oracle database is backed up, you use the RMAN utility. You should not use the RMAN utility to change the retention period. Instead, use the `bkup_api` utility.

The backup feature provided by Database as a Service backs up files and directories specified in configuration files.

You can edit the configuration files to change which files and directories are backed up:

- `/home/oracle/bkup/oscfg.spec`: Lists system files and directories to be backed up
- `/home/oracle/bkup/dbcfg.spec`: Lists Oracle Database files to be backed up, including the file that contains the keystore (wallet), initialization parameter file, and network configuration files

You can use the `bkup_api` utility to specify:

- The number of days backups should be retained
- The number of days in the backup cycle

The backup feature provided by Database as a Service uses the Linux `cron` job scheduler to perform automatic backups. There is an entry in the `/etc/crontab` file specifying when the `bkup_api` utility should run. Edit this entry if you want to change the scheduled time.

Summary

In this lesson, you should have learned how to:

- Create consistent database backups
- Back up your database without shutting it down
- Create incremental backups
- Modify the DBCS default backup configuration
- Create DBCS backups



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 18: Overview

- 18-1: Backing Up the Control File
- 18-2: Verifying Automatic Backups of the Control File and SPFILE
- 18-3: Checking Storage Availability
- 18-4: Creating a Whole Database Backup
- 18-5: Creating a Partial Database Backup



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Performing Database Recovery

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Determine the need for performing recovery
- Describe and use available options, such as Recovery Manager (RMAN) and the Data Recovery Advisor
- Recover data files
- Restore and recover a DBCS database



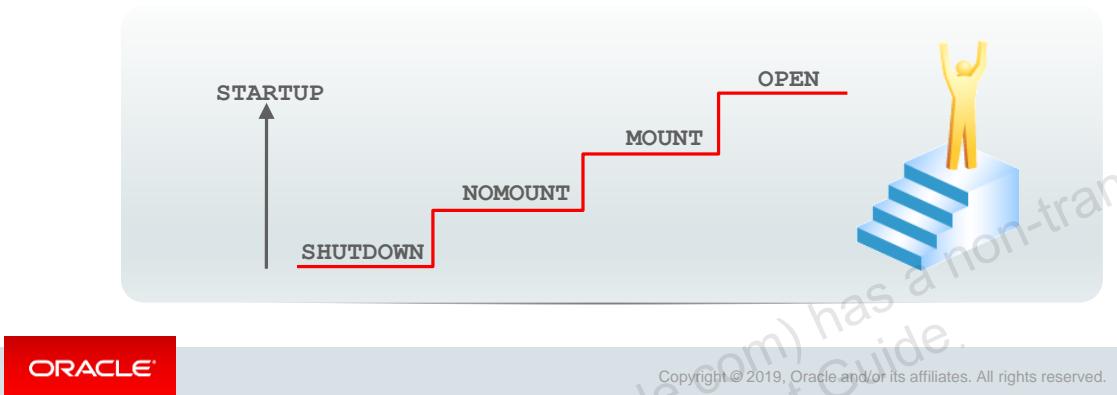
ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Opening a Database

To open a database:

- All control files must be present and synchronized
- All online data files must be present and synchronized
- At least one member of each redo log group must be present



As a database moves from the shutdown stage to being fully open, it performs internal consistency checks during the following stages:

- **NOMOUNT:** For an instance to reach the NOMOUNT (also known as STARTED) status, the instance must read the initialization parameter file. No database files are checked while the instance enters the NOMOUNT state.
- **MOUNT:** As the instance moves to the MOUNT status, it checks whether all control files listed in the initialization parameter file are present and synchronized. If even one control file is missing or corrupt, the instance returns an error (noting the missing control file) to the administrator and remains in the NOMOUNT state.
- **OPEN:** When the instance moves from the MOUNT state to the OPEN state, it does the following:
 - Checks whether all redo log groups known to the control file have at least one member present. Any missing members are noted in the alert log.
 - Verifies that all data files known to the control file are present unless they have been taken offline. Offline files are not checked until the administrator tries to bring them online. The administrator may take a data file offline and open the instance if the data file does not belong to the SYSTEM or UNDO tablespaces. If any files are missing, an error noting the first missing file is returned to the administrator and the instance remains in the MOUNT state. When the instance finds files that are missing, only the first file causing a problem appears in the error message. To find all files that need recovery, the administrator can check the V\$RECOVER_FILE dynamic performance view to get a complete list of the files that need attention:

```
SQL> startup
ORACLE instance started.

Total System Global Area  171966464 bytes
Fixed Size                  775608 bytes
Variable Size                145762888 bytes
Database Buffers              25165824 bytes
Redo Buffers                  262144 bytes

Database mounted.

ORA-01157: cannot identify/lock data file 4 - see DBWR trace file
ORA-01110: data file 4: '/oracle/oradata/orcl/users01.dbf'

SQL> SELECT name, error
  2  FROM v$datafile
  3  JOIN v$recover_file
  4  USING (file#);

NAME                      ERROR
-----
/oracle/oradata/orcl/users01.dbf    FILE NOT FOUND
/oracle/oradata/orcl/example01.dbf FILE NOT FOUND
```

- Verifies that all data files that are not offline or read-only are synchronized with the control file. If necessary, instance recovery is automatically performed. However, if a file is out of synchronization to the extent that it cannot be recovered by using the online redo log groups, then the administrator must perform media recovery. If any files require media recovery, an error message noting the first file requiring recovery is returned to the administrator and the instance remains in the MOUNT state:

```
ORA-01113: file 4 needs media recovery
ORA-01110: data file 4: '/oracle/oradata/orcl/users01.dbf'
```

Again, V\$RECOVER_FILE gives a complete list of files that need attention. Files that are present and require media recovery are listed, but no error message is displayed.

Keeping a Database Open

After the database is open, it fails in case of the loss of:

- Any control file
- A data file belonging to the system or undo tablespaces
- An entire redo log group (as long as at least one member of the group is available, the instance remains open)



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

After a database is open, instance failure can be caused by media failure (for example, by the loss of a control file, the loss of an entire redo log group, or the loss of a data file belonging to the SYSTEM or UNDO tablespaces). Even if an inactive redo log group is lost, the database would eventually fail due to log switches.

In many cases, the failed instance does not completely shut down but is unable to continue to perform work. Recovering from these types of media failure must be done with the database down. As a result, the administrator must use the SHUTDOWN ABORT command before beginning recovery efforts.

The loss of data files belonging to other tablespaces does not cause instance failure, and the database can be recovered while open, with work continuing in other tablespaces.

These errors can be detected by inspecting the alert log file or by using the Data Recovery Advisor.

Data Recovery Advisor

- Fast detection, analysis, and repair of failures
- Handling of down time and run time failures
- Minimizing disruptions for users
- User interfaces:
 - Enterprise Manager Cloud Control
 - RMAN command line



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Data Recovery Advisor automatically gathers data failure information when an error is encountered. In addition, it can proactively check for failures. In this mode, it can potentially detect and analyze data failures before a database process discovers the corruption and signals an error. (Note that repairs are always under human control.)

Data failures can be very serious. For example, if your current log files are missing, you cannot open your database. Some data failures (like block corruptions in data files) are not catastrophic because they do not take the database down or prevent you from opening the Oracle database. The Data Recovery Advisor handles both cases: the one when you cannot start the database (because required database files are missing, inconsistent, or corrupted) and the one when file corruptions are discovered during run time.

The preferred way to address serious data failures is as follows:

1. Fail over to a standby database if you are in a Data Guard configuration. This allows users to come back online as soon as possible.
2. Repair the primary cause of the data failure.

User Interfaces

The Data Recovery Advisor is available in Enterprise Manager Cloud Control. When failures exist, there are several ways to access the Data Recovery Advisor.

You can also use the Data Recovery Advisor by using the RMAN command line:

```
rman target /  
rman> list failure all;
```

Supported Database Configurations

In the current release, the Data Recovery Advisor supports single-instance databases. Oracle Real Application Cluster databases are not supported.

The Data Recovery Advisor cannot use blocks or files transferred from a standby database to repair failures on a primary database. Furthermore, you cannot use the Data Recovery Advisor to diagnose and repair failures on a standby database. However, the Data Recovery Advisor does support failover to a standby database as a repair option (as mentioned previously).

Loss of a Control File

- If a control file is lost or corrupted, the instance normally aborts.
- If control files are stored as regular file system files:
 - Shut down the database
 - Copy an existing control file to replace the lost control file
- After the control file is successfully restored, open the database.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The options for recovery from the loss of a control file depend on the storage configuration of the control files and on whether at least one control file remains or have all been lost.

If your control files are stored as regular file system files and at least one control file copy remains, then, while the database is down, you can just copy one of the remaining control files to the missing file's location. If the media failure is due to the loss of a disk drive or controller, copy one of the remaining control files to some other location and update the instance's parameter file to point to the new location. Alternatively, you can delete the reference to the missing control file from the initialization parameter file. Remember that Oracle recommends having at least two control files at all times.

If you are using ASM storage, and at least one control file copy remains, you can perform guided recovery by using Enterprise Manager Cloud Control or perform manual recovery by using RMAN as follows:

1. Put the database in NOMOUNT mode.
2. Connect to RMAN and issue the RESTORE CONTROLFILE command to restore the control file from an existing control file. For example:

```
restore controlfile from
'+DATA/orcl/controlfile/current.260.695209463';
```
3. After the control file is successfully restored, open the database.

Note: Recovering from the loss of all control files is covered in the course titled *Oracle Database 18c: Backup and Recovery Workshop*.

Loss of a Redo Log File

If a member of a redo log file group is lost and if the group still has at least one member, note the following results:

- Normal operation of the instance is not affected.
- You receive a message in the alert log notifying you that a member cannot be found.
- You can restore the missing log file by dropping the lost redo log member and adding a new member.
- If the group with the missing log file has been archived, you can clear the log group to re-create the missing file.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Recovering from the loss of a single redo log group member should not affect the running instance.

To perform this recovery by using SQL commands:

1. Determine whether there is a missing log file by examining the alert log.
2. Restore the missing file by first dropping the lost redo log member:

```
ALTER DATABASE DROP LOGFILE MEMBER '<filename>'
```

Then add a new member to replace the lost redo log member:

```
ALTER DATABASE ADD LOGFILE MEMBER '<filename>'  
TO GROUP <integer>
```

Note: If you are using Oracle Managed Files (OMF) for your redo log files and you use the preceding syntax to add a new redo log member to an existing group, that new redo log member file will not be an OMF file. If you want to ensure that the new redo log member is an OMF file, then the easiest recovery option would be to create a new redo log group and then drop the redo log group that had the missing redo log member.

3. If the media failure is due to the loss of a disk drive or controller, rename the missing file.
4. If the group has already been archived or if you are in NOARCHIVELOG mode, you may choose to solve the problem by clearing the log group to re-create the missing file or files. You can clear the affected group manually with the following command:

```
ALTER DATABASE CLEAR LOGFILE GROUP <integer>;
```

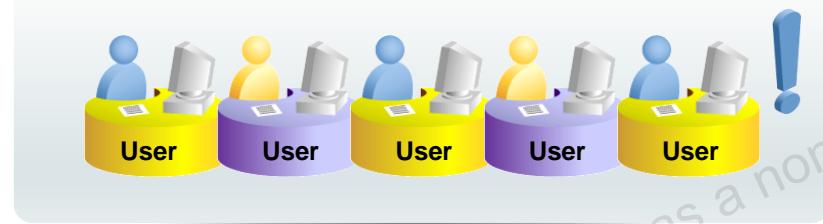
Note: Enterprise Manager does not allow you to clear a log group that has not been archived. Doing so breaks the chain of redo information. If you must clear an unarchived log group, you should *immediately* take a full backup of the whole database. Failure to do so may result in a loss of data if another failure occurs. To clear an unarchived log group, use the following command:

```
ALTER DATABASE CLEAR UNARCHIVED LOGFILE GROUP <integer>
```

Loss of a Data File in NOARCHIVELOG Mode

If the database is in `NOARCHIVELOG` mode and if any data file is lost, perform the following tasks:

1. Shut down the instance if it is not already down.
2. Restore the entire database, including all data and control files, from the backup.
3. Open the database.
4. Have users re-enter all the changes that were made since the last backup.



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The loss of *any* data file from a database in `NOARCHIVELOG` mode requires complete restoration of the database, including control files and all data files.

With the database in `NOARCHIVELOG` mode, recovery is possible only up to the time of the last backup. So users must re-enter all changes made since that backup.

If you have a database in `NOARCHIVELOG` mode that has an incremental backup strategy, RMAN first restores the most recent level 0 and then RMAN recovery applies the incremental backups.

Loss of a Noncritical Data File in ARCHIVELOG Mode

- If a data file is lost or corrupted, and if that file does not belong to the SYSTEM or UNDO tablespace, you restore and recover the missing data file.
- The other database files are available for users.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

With the database in ARCHIVELOG mode, the loss of any data file not belonging to the SYSTEM or UNDO tablespaces affects only the objects that are in the missing file. The rest of the database remains available for users to continue work.

Because the database is in ARCHIVELOG mode, recovery is possible up to the time of the last commit, and users are not required to re-enter any data.

Loss of a System-Critical Data File in ARCHIVELOG Mode

If a data file is lost or corrupted, and if that file belongs to the SYSTEM or UNDO tablespace, perform the following tasks:

1. The instance may or may not shut down automatically. If it does not, use SHUTDOWN ABORT to bring the instance down.
2. Mount the database.
3. Restore and recover the missing data file.
4. Open the database.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Data files belonging to the SYSTEM tablespace or containing UNDO data are considered system critical. A loss of one of these files requires the database to be restored from the MOUNT state (unlike other data files that may be restored with the database open).

DBCS: Performing Recovery by Using the Console



- Restore from the most recent backup and perform complete recovery.
- Restore from a backup and recover to a specific data and time.
- Restore from a backup and recover to a specific system change number (SCN).

ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can use the Database Cloud Service console to perform recovery as described in the slide. Note that you must have configured backups through Database Cloud Service or taken an on-demand backup through the DBCS console to use the console to perform recovery operations. See *Administering Oracle Database Cloud Service* for additional details.

DBCS: Performing Recovery by Using the `dbaascli` Utility



Use the `orec` subcommand of the `dbaascli` utility to restore and recover the database:

- Restoring from the most recent backup and performing complete recovery

```
# dbaascli orec --args -latest
```
- Restoring from a specific backup and performing point-in-time recovery

```
# dbaascli orec --args -pitr backup-tag
```
- Restoring from the most recent backup and performing recovery through the specified system change number (SCN)

```
# dbaascli orec --args -scn SCN
```
- Restoring from a specific long-term backup and performing point-in-time

```
# dbaascli orec --args -keep -tag backup-tag
```



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

To restore from a backup and perform recovery on a Database Cloud Service deployment hosting a single-instance database, you use the `orec` subcommand of the `dbaascli` utility, which is available on the compute node.

The `orec` subcommand must be run with `root` access. Therefore, you need to connect to the compute node as the `opc` user to perform recovery operations.

You can restore from the most recent backup, a specific backup, or a specific long-term backup.

Use the following command to obtain a list of backups, including the backup tag for each:

```
# dbaascli orec --args -list
```

Use this command to obtain a list of long-term backups, including the backup tag for each:

```
# dbaascli orec --args -keep -list
```

Summary

In this lesson, you should have learned how to:

- Determine the need for performing recovery
- Describe and use available options, such as Recovery Manager (RMAN) and the Data Recovery Advisor
- Recover data files
- Restore and recover a DBCS database



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 19: Overview

- 19-1: Recovering from an Essential Data File Loss
- 19-2: Recovering from an Application Data File Loss



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Monitoring and Tuning Database Performance

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

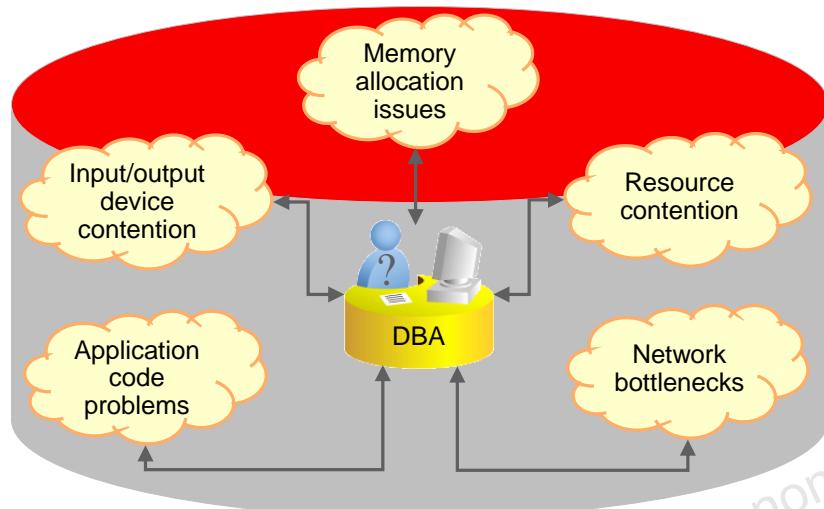
- Describe the activities that you perform to manage database performance
- Use Enterprise Manager Database Express and performance views to monitor database instance performance
- Describe the Oracle performance tuning methodology
- Describe the server statistics and metrics that are collected by the Oracle Database server
- Configure and monitor memory components for optimal performance



ORACLE®

Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Performance Management Activities



ORACLE

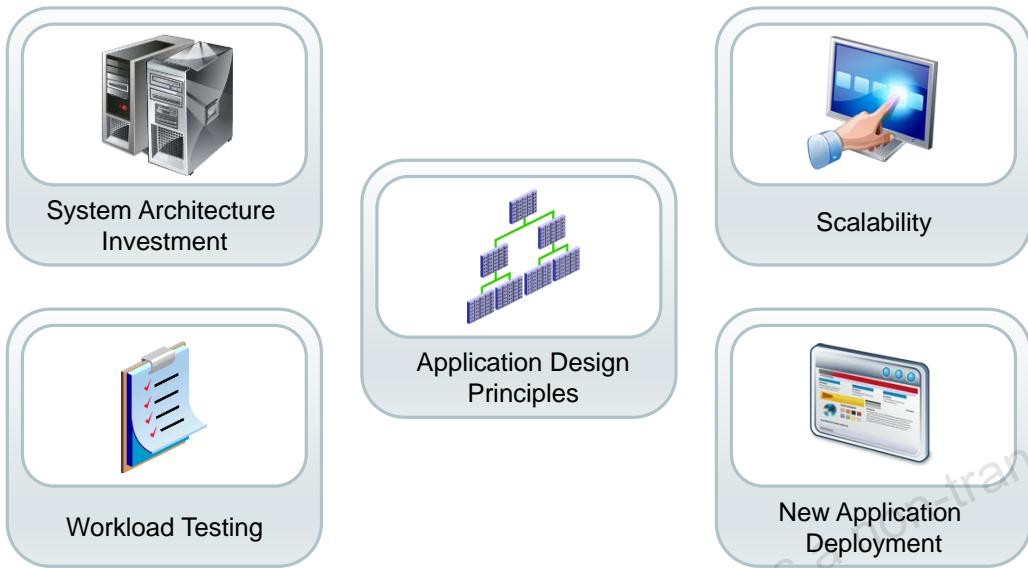
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Performance management includes the following activities:

- Performance planning is the process of establishing the environment: the hardware, software, operating system, network infrastructure, and so on.
- Performance monitoring is an activity that helps the DBA locate bottlenecks and correct problem areas.
- Instance tuning is the actual adjustment of Oracle Database parameters and operating system (OS) parameters to gain better performance of the Oracle Database.
- SQL tuning involves making your application submit efficient SQL statements. SQL tuning is performed for the application as a whole, as well as for individual statements. At the application level, you want to be sure that different parts of the application are taking advantage of each other's work and are not competing for resources unnecessarily.

A DBA can look at hundreds of performance measurements, covering everything from network performance and disk input/output (I/O) speed to the time spent working on individual application operations. These performance measurements are commonly referred to as database metrics.

Performance Planning Considerations



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

There are many facets to performance planning. Planning must include a balance between performance (speed), cost, and reliability.

Investment in System Architecture: You must consider the investment in your system architecture—the hardware and software infrastructure needed to meet your requirements. This, of course, requires analysis to determine the value for your given environment, application, and performance requirements. For example, the number of hard drives and controllers has an impact on the speed of data access.

Scalability: The ability of an application to scale is also important. This means that you are able to handle more and more users, clients, sessions, or transactions without incurring a huge impact on overall system performance. The most obvious violator of scalability is serializing operations among users. If all users go through a single path one at a time, then, as more users are added, there are definitely adverse effects on performance. This is because more and more users line up to go through that path. Poorly written SQL also affects scalability. It requires many users to wait for inefficient SQL to complete, each user competing with the other on a large number of resources that they are not actually in need of.

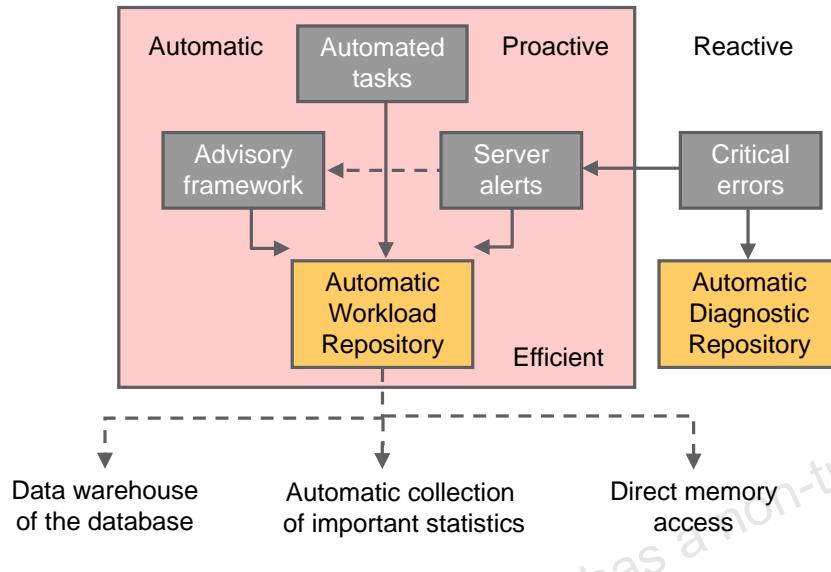
Application Design Principles: The principles of application design can greatly affect performance. Simplicity of design, use of views and indexes, and data modeling are all very important.

Workload Testing: Any application must be tested under a representative production workload. This requires estimating database size and workload and generating test data and system load.

Deployment of New Applications: Performance must be considered as new applications (or new versions of applications) are deployed. Sometimes, design decisions are made to maintain compatibility with old systems during the rollout. A new database should be configured (on the basis of the production environment) specifically for the applications that it hosts.

A difficult and necessary task is testing the existing applications when changing the infrastructure (for example, upgrading the database to a newer version or changing the operating system or server hardware). Before the application is deployed for production in the new configuration, you want to know the impact. The application will almost certainly require additional tuning. You need to know that the critical functionality will perform, without errors.

Database Maintenance



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Proactive database maintenance is made easy by the sophisticated infrastructure of the Oracle Database, including the following main elements:

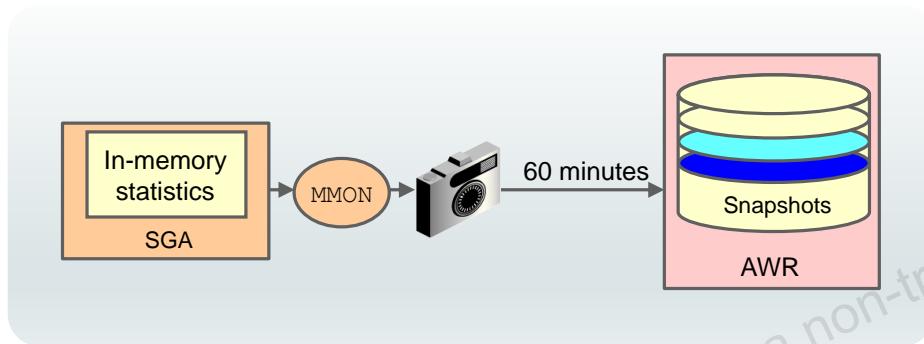
- The Automatic Workload Repository (AWR) is a built-in repository in each Oracle database. At regular intervals, the Oracle Database server takes a snapshot of all its vital statistics and workload information and stores this data in the AWR. The captured data can be analyzed by you, by the database server itself, or by both.
- Using automated tasks, the database server performs routine maintenance operations, such as regular backups, refreshing optimizer statistics, and database health checks.

Reactive database maintenance includes critical errors and conditions discovered by database health checkers:

- For problems that cannot be resolved automatically and require administrators to be notified (such as running out of space), the Oracle Database server provides server-generated alerts. The Oracle Database server, by default, monitors itself and sends out alerts to notify you of problems. The alerts notify you and often also provide recommendations on how to resolve the reported problem. The DBA can also be alerted by users whose transactions are locked by other users' transactions and are waiting for locks to be released.
- Recommendations are generated from several advisors, each of which is responsible for a subsystem. For example, there are memory, segment, and SQL advisors.

Automatic Workload Repository (AWR)

- Built-in repository of performance information
- Snapshots of database metrics taken every 60 minutes and retained for eight days
- Foundation for all self-management functions



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

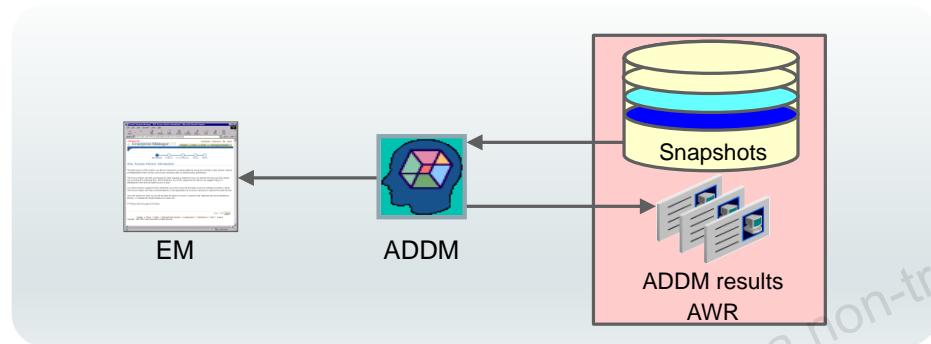
The Automatic Workload Repository (AWR) is the infrastructure that provides services to Oracle Database components to collect, maintain, and use statistics for problem detection and self-tuning purposes. You can view it as a data warehouse for database statistics, metrics, and so on.

Every 60 minutes (by default), the database automatically captures statistical information from the SGA and stores it in the AWR in the form of snapshots. These snapshots are stored on disk by a background process called Manageability Monitor (MMON). By default, snapshots are retained for eight days. You can modify both the snapshot interval and the retention intervals.

The AWR contains hundreds of tables, all belonging to the `SYS` schema and stored in the `SYSAUX` tablespace. Oracle recommends that the repository be accessed only through Enterprise Manager or the `DBMS_WORKLOAD_REPOSITORY` package to work with the AWR. Direct data manipulation language (DML) commands against the repository tables are not supported.

Automatic Database Diagnostic Monitor (ADDM)

- Runs after each AWR snapshot
- Monitors the instance; detects bottlenecks
- Stores results in the AWR



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Unlike the other advisors, the ADDM runs automatically after each AWR snapshot. Each time a snapshot is taken, the ADDM performs an analysis of the period corresponding to the last two snapshots. The ADDM proactively monitors the instance and detects most bottlenecks before they become a significant problem.

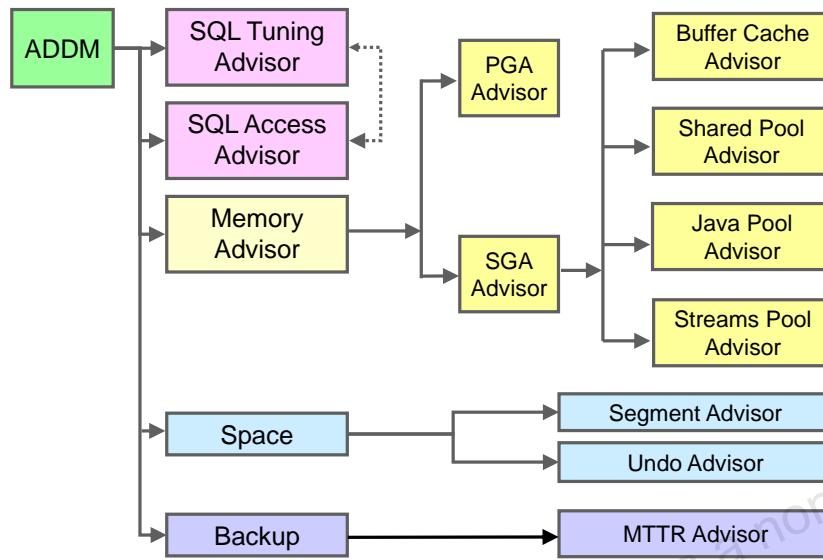
In many cases, the ADDM recommends solutions for detected problems and even quantifies the benefits for the recommendations.

Some common problems that are detected by the ADDM:

- CPU bottlenecks
- Poor Oracle Net connection management
- Lock contention
- Input/output (I/O) capacity
- Undersized database instance memory structures
- High-load SQL statements

The results of each ADDM analysis are stored in the AWR and are also accessible through Enterprise Manager.

Advisory Framework



Advisors provide you with useful feedback about resource utilization and performance for their respective server components. For example, Memory Advisor provides a recommended value for the `MEMORY_TARGET` initialization parameter, which controls the total amount of memory used by the Oracle database instance.

By building on the data captured in the AWR, the ADDM enables the Oracle Database server to diagnose its own performance and determine how identified problems can be resolved. ADDM runs automatically after each AWR statistics capture. It can potentially call other advisors.

Here are the major benefits that are provided by the advisor infrastructure:

- All advisors use a uniform interface.
- All advisors have a common data source and results storage by using the workload repository.

Not all advisors are shown in the slide (for example, Data Recovery Advisor and SQL Repair Advisor are not listed).

Automatic Database Diagnostic Monitor (ADDM)

The ADDM is a server-based expert that reviews database performance every 60 minutes. Its goal is to detect possible system bottlenecks early and recommend fixes before system performance degrades noticeably.



Memory Advisors

Memory Advisor is actually a collection of several advisory functions that help determine the best settings for the total memory used by the database instance. The System Global Area (SGA) has a set of advisors for the shared pool, database buffer cache, Java pool, and streams pool. The Java pool and streams pool advisors are not exposed on the Enterprise Manager Memory Advisor page. There is an advisor for the Program Global Area (PGA). In addition to the advisory functions, this advisor provides a central point of control for the large pool and the Java pool.

Mean-Time-To-Recover (MTTR) Advisor

Using MTTR Advisor, you set the length of time required for the database to recover after an instance crash.

Segment Advisor

This advisor looks for tables and indexes that consume more space than they require. The advisor checks for inefficient space consumption at the tablespace or schema level and produces scripts to reduce space consumption where possible.

SQL Access Advisor

This advisor analyzes all SQL statements that are issued in a given period and suggests the creation of additional indexes or materialized views that will improve performance.

SQL Tuning Advisor

This advisor analyzes an individual SQL statement and makes recommendations for improving its performance. Recommendations may include actions, such as rewriting the statement, changing the instance configuration, or adding indexes.

Undo Management Advisor

With Undo Management Advisor, you can determine the undo tablespace size that is required to support a given retention period. Undo management and the use of the advisor is covered in the lesson titled “Managing Undo Data.”

Data Recovery Advisor

This advisor automatically diagnoses persistent data failures, presents repair options to the user, and executes repairs at the user’s request. The purpose of Data Recovery Advisor is to reduce the mean time to recover (MTTR) and provide a centralized tool for automated data repair.

SQL Repair Advisor

You run SQL Repair Advisor after a SQL statement fails with a critical error that generates a problem in the Automatic Diagnostic Repository. The advisor analyzes the statement and, in many cases, recommends a patch to repair the statement. If you implement the recommendation, the applied SQL patch circumvents the failure by causing the query optimizer to choose an alternative execution plan for future executions. This is done without changing the SQL statement itself.

Automated Maintenance Tasks

- Autotask maintenance process:
 1. Maintenance window opens.
 2. Autotask background process schedules jobs.
 3. Scheduler initiates jobs.
 4. Resource Manager limits the impact of Autotask jobs.
- Default Autotask maintenance jobs:
 - Gathering optimizer statistics
 - Automatic Segment Advisor
 - Automatic SQL Advisor



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

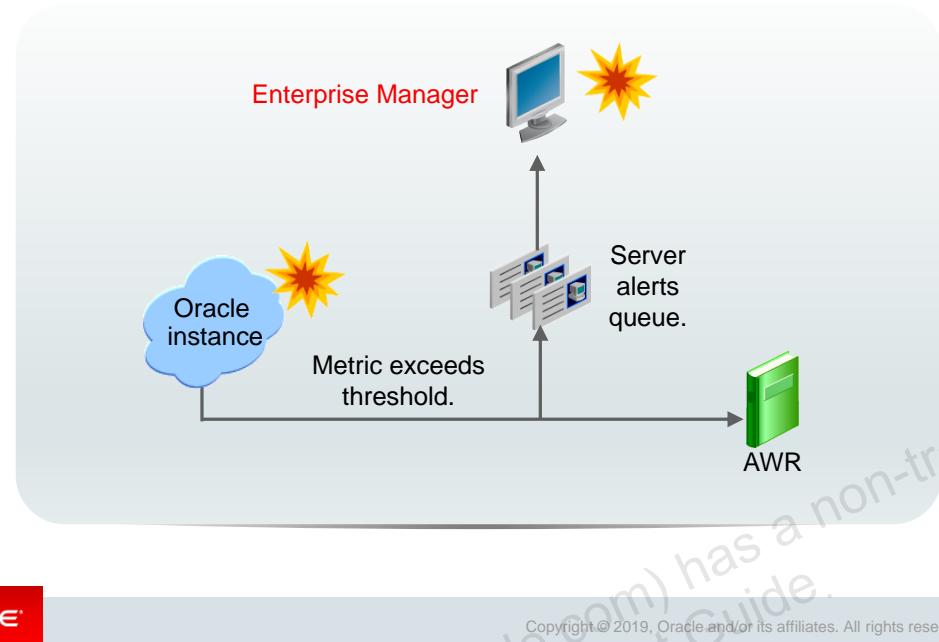
By analyzing the information stored in the AWR, the database server can identify the need to perform routine maintenance tasks, such as optimizer statistics refresh. The automated maintenance tasks infrastructure enables the Oracle Database server to automatically perform such operations. It uses the Scheduler to run such tasks in predefined maintenance windows.

By default, the weekday maintenance windows start at 10:00 PM and last four hours. On Saturday and Sunday, the maintenance window starts at 6:00 AM and lasts for 20 hours. All attributes of the maintenance windows are customizable, including the start and end times, frequency, days of the week, and so on. In addition, the impact of automated maintenance tasks on normal database operations can be limited by associating a Database Resource Manager resource plan to the maintenance window.

Examples of maintenance:

- Optimizer statistics are automatically refreshed by using the automatic maintenance task infrastructure.
- Automatic Segment Advisor has default jobs, which run in the maintenance window.
- When creating a database with the DBCA, you can initiate regular database backups.

Server-Generated Alerts



Alerts are notifications of when a database is in an undesirable state and needs your attention. By default, the Oracle Database server provides alerts via Enterprise Manager. Optionally, Enterprise Manager can be configured to send an email message to the administrator about problem conditions as well as display alert information on the console.

You can also set thresholds on many of the pertinent metrics for your system. Oracle Database proactively notifies you if the database deviates sufficiently from normal readings to reach those thresholds. An early notification of potential problems enables you to respond quickly and, in many cases, resolve issues before users even notice them.

Approximately 60 metrics are monitored by default, among which are:

- Broken Job Count
- Database Time Spent Waiting (%)
- Dump Area Used (%)
- SQL Response Time (%) Compared to Baseline
- Tablespace Used (%)
- Generic Incident

A few additional key metrics can provide early problem notification:

- Average File Read Time (centiseconds)
- Response Time (per transaction)
- Wait Time (%)

Setting Metric Thresholds

Metric	Comparison Operator	Warning Threshold	Critical Threshold	Corrective Actions	Collection Schedule	Edit
Archiver Hung Alert Log Error	Contains	<input type="text"/>	ORA-	None		
Data Block Corruption Alert Log Error	Contains	<input type="text"/>	ORA-	None		
Generic Alert Log Error	Matches	<input type="text"/> ORA-0*(600?	<input type="text"/>	None		
Media Failure Alert Log Error	Contains	<input type="text"/>	ORA-	None		
Session Terminated Alert Log Error	Contains	<input type="text"/> ORA-	<input type="text"/>	None		
Archiver Hung Alert Log Error Status	>	<input type="text"/> 0	<input type="text"/>	None		
Data Block Corruption Alert Log Error Status	>	<input type="text"/> 0	<input type="text"/>	None		



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

To access the Metric and Collection Settings page, expand the Oracle Database menu and select Metric and Collection Settings from the Monitoring submenu.

Enter your desired warning and critical threshold values for the metric. The appropriate alerts appear when the database reaches your specified values.

The thresholds that are already set appear in the “Metrics with thresholds” list. By default, approximately 60 metrics have preset thresholds; you may change these as needed. The “All metrics” list shows the metrics that do not have thresholds set.

Click the Edit icon to access a page where you can specify additional corrective actions for either warning or critical thresholds.

Click a Collection Schedule link to change the scheduled collection interval. Be aware that each schedule affects a group of metrics.

Reacting to Alerts

- If necessary, you should gather more input (for example, by running ADDM or another advisor).
- Investigate critical errors.
- Take corrective measures.
- Acknowledge alerts that are not automatically cleared.



ORACLE

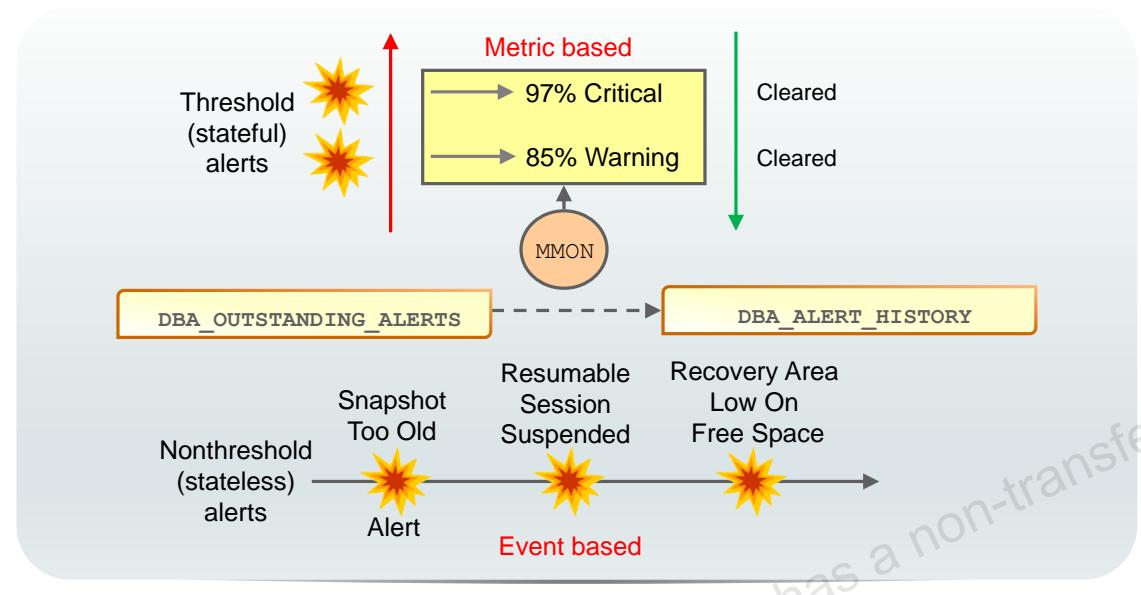
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

When you receive an alert, follow the recommendations that it provides. You can also consider running the ADDM (or another advisor as appropriate) to obtain more detailed diagnostics of system or object behavior.

Alerts and incidents are generated for critical errors. Critical errors usually generate incidents that are collected into problems. You use the Support Workbench to investigate and possibly report the problem to Oracle Support.

Most alerts (such as “Out of Space”) are cleared automatically when the cause of the problem disappears. However, other alerts (such as Generic Alert Log Error) are sent to you for notification and must be acknowledged by you. After taking the necessary corrective measures, you acknowledge an alert by clearing or purging it. Clearing an alert sends the alert to the Alert History, which is accessible from the Monitoring submenu. Purging an alert removes it from the Alert History.

Alert Types and Clearing Alerts



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

There are two kinds of server-generated alerts: threshold and nonthreshold.

Most server-generated alerts are configured by setting a warning and critical threshold values on database metrics. You can define thresholds for more than 120 metrics, including the following:

- Physical Reads Per Sec
- User Commits Per Sec
- SQL Service Response Time

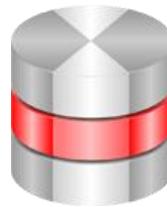
Except for the Tablespace Space Usage metric, which is database related, the other metrics are instance related. Threshold alerts are also referred to as *stateful alerts*, which are automatically cleared when an alert condition clears. Stateful alerts appear in `DBA_OUTSTANDING_ALERTS` and, when cleared, go to `DBA_ALERT_HISTORY`.

Other server-generated alerts correspond to specific database events, such as `ORA-*` errors, "Snapshot too old" errors, Recovery Area Low On Free Space, and Resumable Session Suspended. These are non-threshold-based alerts, also referred to as *stateless alerts*. Stateless alerts go directly to the History table.

Database Server Statistics and Metrics

Cumulative statistics:

- Wait events with time information
- Time model



Metrics: Statistic rates



Sampled statistics:

- Active session history
- Statistics by session, SQL, and service
- Other dimensions



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Oracle Database server software captures information about its own operation. Three major types of data are collected: cumulative statistics, metrics, and sampled statistics.

Cumulative statistics are counts and timing information of a variety of events that occur in the database server. Some are quite important, such as buffer busy waits. Others have little impact on tuning, such as index block split. The most important events for tuning are usually the ones showing the greatest cumulative time values. The statistics in Oracle Database are correlated by the use of a time model. The time model statistics are based on a percentage of DB time, giving them a common basis for comparison.

Metrics are statistic counts per unit. The unit could be time (such as seconds), transaction, or session. Metrics provide a base to proactively monitor performance. You can set thresholds on a metric, causing an alert to be generated. For example, you can set thresholds for when the reads per millisecond exceed a previously recorded peak value or when the archive log area is 95% full.

Sampled statistics are gathered automatically when `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`. Sampled statistics allow you to look back in time. You can view session and system statistics that were gathered in the past, in various dimensions, even if you had not thought of specifying data collection for these beforehand.

Performance Monitoring

- Enterprise Manager Database Express
- Enterprise Manager Cloud Control
- Performance views

Instance/Database

V\$DATABASE
V\$INSTANCE
V\$PARAMETER
V\$SPPARAMETER
V\$SYSTEM_PARAMETER
V\$PROCESS
V\$BGPROCESS
V\$PX_PROCESS_SYSSTAT
V\$SYSTEM_EVENT

Disk

V\$DATAFILE
V\$FILESTAT
V\$LOG
V\$LOG_HISTORY
V\$DBFILE
V\$TEMPFILE
V\$TEMPSEG_USAGE
V\$SEGMENT_STATISTICS

Memory

V\$BUFFER_POOL_STATISTICS
V\$LIBRARYCACHE
V\$SGAINFO
V\$PGASTAT

Contention

V\$LOCK
V\$UNDOSTAT
V\$WAITSTAT
V\$LATCH



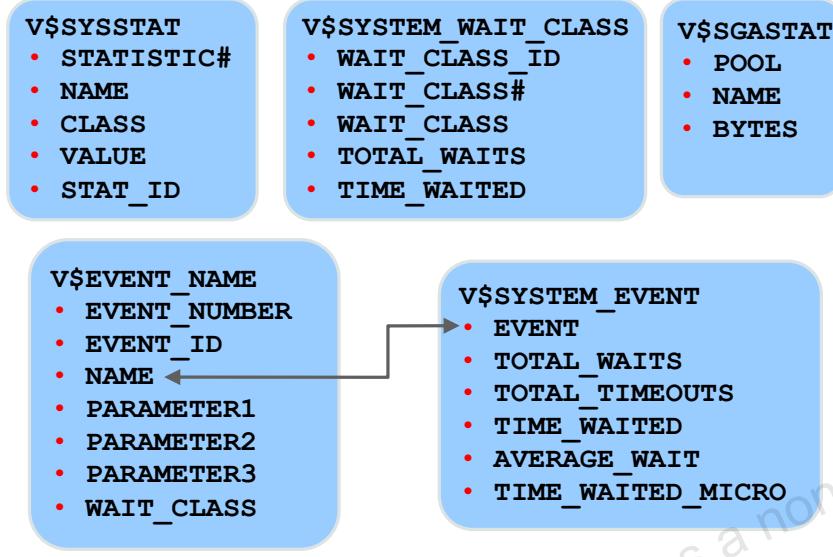
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can respond to changes in performance only if you know the performance has changed. Oracle Database provides several ways to monitor the current performance of the database instance.

- **Enterprise Manager Database Express:** The database home page provides a quick check of the health of the instance and the server, with graphs showing CPU usage, active sessions, memory, and data storage usage. The home page also shows any alerts that have been triggered. Additional details are available on the Performance Hub page. As mentioned earlier in the lesson, the ADDM analysis results are accessible through Enterprise Manager.
- **Enterprise Manager Cloud Control:** Enterprise Manager Cloud Control also provides performance monitoring capabilities.
- **Performance views:** You can access these views directly with SQL*Plus. Occasionally, you may need to access these views for some details about the raw statistics.

See *Oracle Database Performance Tuning Guide* and *Oracle Database Reference* for details and examples.

Viewing Statistics Information



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

To effectively diagnose performance problems, statistics must be available. The Oracle Database instance generates many types of cumulative statistics for the system, sessions, and individual SQL statements at the instance level. The Oracle Database server also tracks cumulative statistics on segments and services. When analyzing a performance problem in any of these scopes, you typically look at the change in statistics (delta value) over the period of time you are interested in.

Note: Instance statistics are dynamic and are reset at every instance startup. These statistics can be captured at a point in time and held in the database in the form of snapshots.

Wait Event Statistics

All the possible wait events are cataloged in the V\$EVENT_NAME view.

Cumulative statistics for all sessions are stored in V\$SYSTEM_EVENT, which shows the total waits for a particular event since instance startup.

When you are troubleshooting, you need to know whether a process has waited for any resource.

System-Wide Statistics

All the system-wide statistics are cataloged in the V\$STATNAME view. Over 400 statistics are available in Oracle Database.

The server displays all calculated system statistics in the V\$SYSSTAT view. You can query this view to find cumulative totals since the instance started.

System-wide statistics are classified by the tuning topic and the debugging purpose. The classes include general instance activity, redo log buffer activity, locking, database buffer cache activity, and so on. Each of the system statistics can belong to more than one class, so you cannot do a simple join on V\$SYSSTATS.CLASS and V\$SYSTEM_WAIT_CLASS.WAIT_CLASS#.

You can also view all wait events for a particular wait class by querying V\$SYSTEM_WAIT_CLASS.

SGA Global Statistics

The server displays all calculated memory statistics in the V\$SGASTAT view. You can query this view to find cumulative totals of detailed SGA usage since the instance started.

When the STATISTICS_LEVEL parameter is set to BASIC, the value of the TIMED_STATISTICS parameter defaults to FALSE. Timing information is not collected for wait events, and much of the performance-monitoring capability of the database is disabled. The explicit setting of TIMED_STATISTICS overrides the value derived from STATISTICS_LEVEL.

Monitoring Wait Events



Wait events: Statistics indicating the server process had to wait for an event to complete

V\$EVENT_NAME

ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

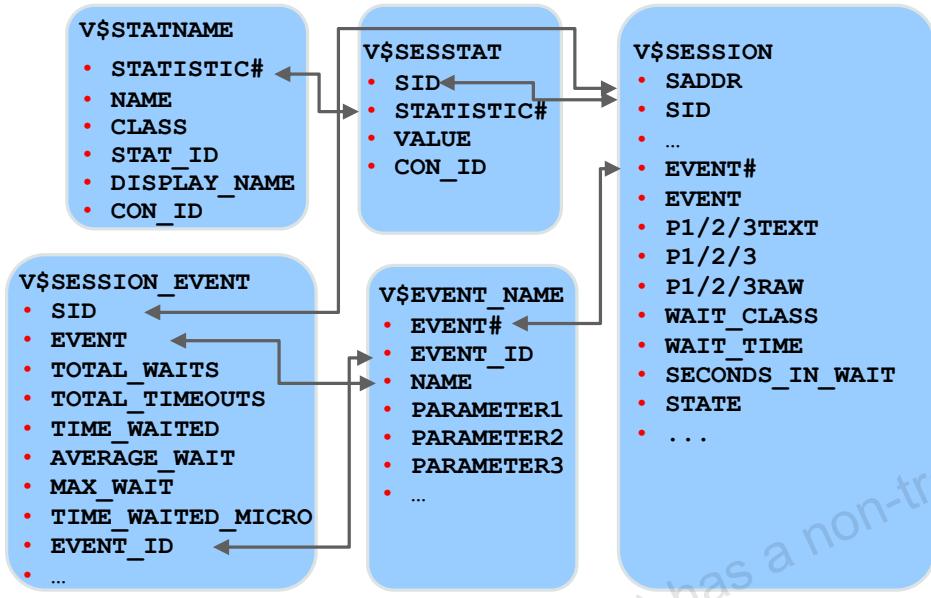
Wait events are statistics that are incremented by a server process or thread to indicate that it had to wait for an event to complete before being able to continue processing. Wait event data reveals various symptoms of problems that might be impacting performance, such as latch contention, buffer contention, and I/O contention. Remember that these are only symptoms of problems, not the actual causes.

A collection of wait events provides information about the sessions or processes that had to wait or must wait for different reasons.

Wait events are grouped into classes. The wait event classes include: Administrative, Application, Cluster, Commit, Concurrency, Configuration, Idle, Network, Other, Scheduler, System I/O, and User I/O.

Wait events are listed in the V\$EVENT_NAME view. There are more than 800 wait events in the Oracle Database, including free buffer wait, latch free, buffer busy waits, DB file sequential read, and DB file scattered read.

Monitoring Sessions



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can display current session information for each user logged on by querying V\$SESSION. For example, you can use V\$SESSION to determine whether a session represents a user session, or was created by a database server process (background).

You can query either V\$SESSION or V\$SESSION_WAIT to determine the resources or events for which active sessions are waiting.

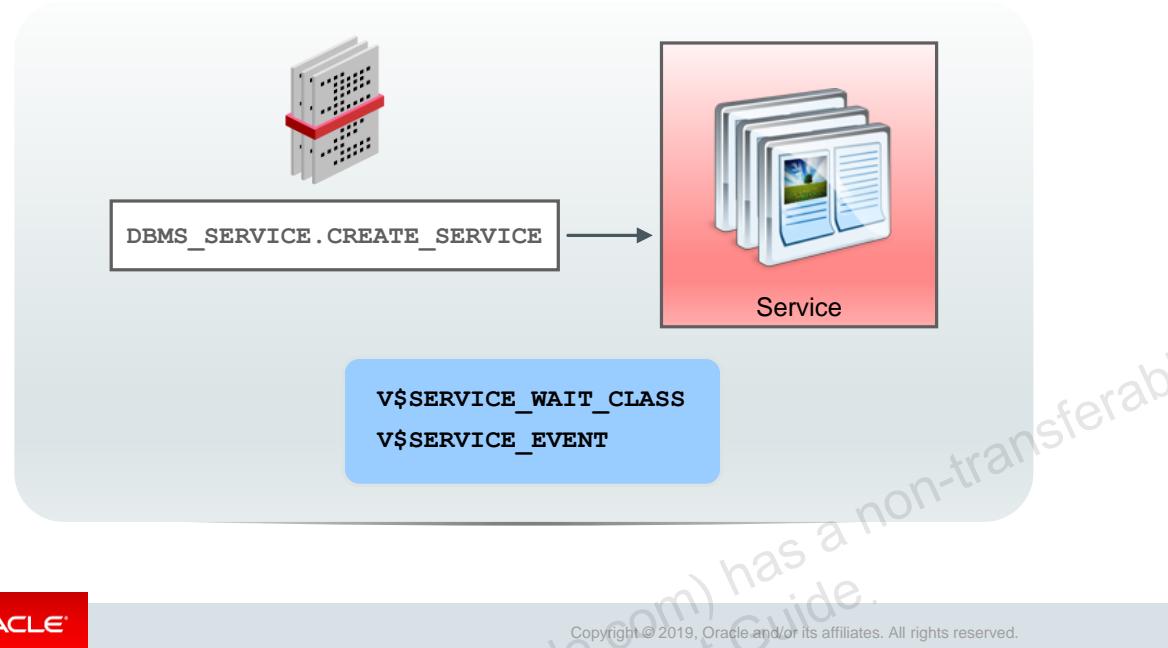
You can view user session statistics in V\$SESSTAT. The V\$SESSION_EVENT view lists information about waits for an event by session.

Cumulative values for instance statistics are generally available through dynamic performance views, such as V\$SESSTAT and V\$SYSSSTAT. Note that the cumulative values in dynamic views are reset when the database instance is shut down.

The V\$MYSTAT view displays the statistics of the current session.

You can also query V\$SESSMETRIC to display the performance metric values for all active sessions. This view lists performance metrics, such as CPU usage, number of physical reads, number of hard parses, and the logical read ratio.

Monitoring Services



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

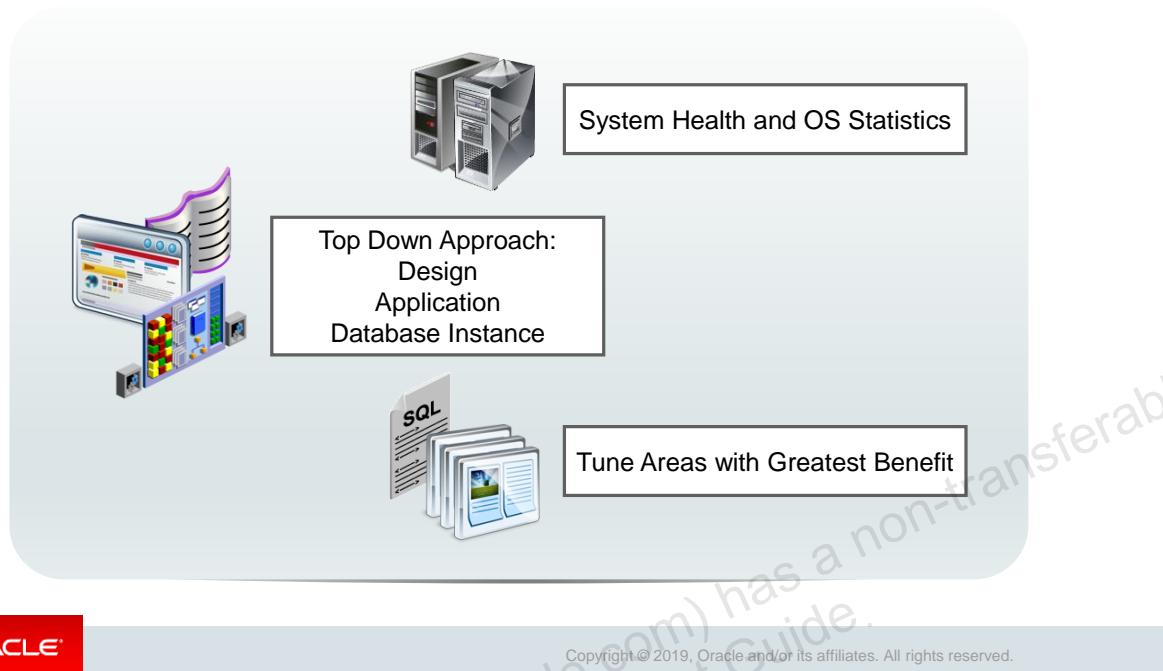
In a multi-tier environment where there is an application server that is pooling database connections, viewing sessions may not provide the information you need to analyze performance. Grouping sessions into service names enables you to monitor performance more accurately. Regardless of the session that was used for a particular request, if it connected via one of these services, the performance data of the session is captured under that service name.

You can define a service in the database by using the `DBMS_SERVICE` package and can use the net service name to assign applications to a service.

Two views provide the same information that their like-named session counterparts provide, except that the information is presented at the service level rather than at the session level.

- `V$SERVICE_WAIT_CLASS` shows wait statistics for each service, broken down by wait class.
- `V$SERVICE_EVENT` shows the same information as `V$SERVICE_WAIT_CLASS`, except that it is further broken down by event ID.

Performance Tuning Methodology

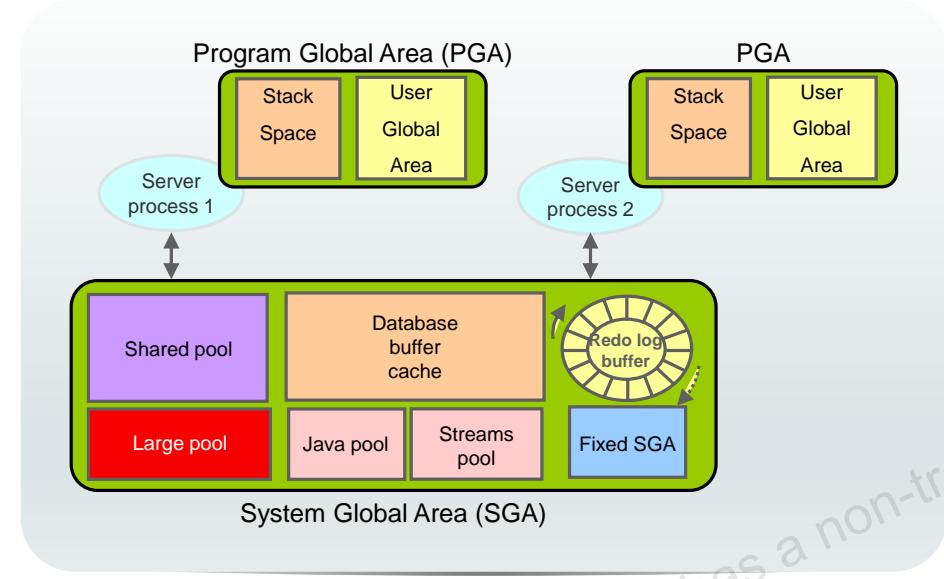


Oracle has developed a tuning methodology based on years of experience. The basic steps are:

1. Check the OS statistics and general machine health before tuning the instance to be sure that the problem is in the database.
2. Tune from the top down. Start with the design, then the application, and then the instance. For example, try to eliminate full table scans that cause I/O contention before tuning the tablespace layout on disk. This activity often requires access to the application code.
3. Tune the area with the greatest potential benefit. The tuning methodology presented in this course is simple. Identify the biggest bottleneck and tune it. Repeat this step. All the various tuning tools have some way to identify the SQL statements, resource contention, or services that are taking the most time. The Oracle database provides a time model and metrics to automate the process of identifying bottlenecks. The advisors available in Oracle Database use this methodology.
4. Stop tuning when you meet your goal. This step implies that you set tuning goals.

This is a general approach to tuning the database instance and may require multiple passes.

Managing Memory Components



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Because there is a finite amount of memory available on a database server and an Oracle Database instance, you must pay attention to how memory is allocated. If too much memory is allowed to be used by a particular area that does not need it, other areas may not function properly because of lack of memory. With the ability to have memory allocation automatically determined and maintained for you, the task is simplified greatly. But even automatically tuned memory needs to be monitored for optimization and may need to be manually configured to some extent.

Guidelines for efficient memory usage include the following.

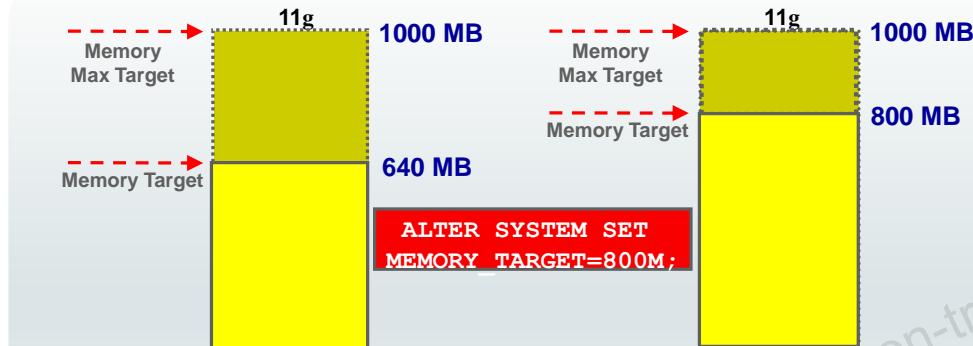
- **Fit the SGA into physical memory:** If possible, it is best to fit the SGA into physical memory, which provides the fastest access. Even though the OS may provide additional virtual memory, this memory, by its nature, can often be swapped out to disk. On some platforms, you can use the `LOCK_SGA` initialization parameter to lock the SGA into physical memory. This parameter cannot be used in conjunction with Automatic Memory Management (AMM) or Automatic Shared Memory Management (ASMM).
- **Tune for the most efficient use of memory:** When a SQL statement executes, data blocks are requested for reading or writing, or both. This is considered a logical I/O. As the block is requested, it is checked to see whether it already exists in memory. If it is not in memory, it is read from disk, which is called a physical I/O. When the block is found in memory, the cost is several orders of magnitude less than the cost of reading the block from disk. The size of the SGA components in combination with the workload has a large effect on the number of physical reads. A simple view of this implies that you should increase the memory for the SGA components as much as possible. A larger SGA is not always better. There is a point where adding more memory yields diminishing returns. This principle applies to the buffer cache, the shared pool, and other SGA components. In practice, you find that

shifting memory from one SGA component to another may increase overall performance, without changing the total amount of memory given to the SGA depending on the characteristics of the workload. Memory has an upper limit in all current machines. That limit may be imposed by the hardware, operating system, or the cost of the memory. The goal of memory tuning is to produce the most efficient use of existing memory. When the workload changes often, the most efficient division of memory between the SGA components will change. There is also the amount of memory allocated to SGA and PGA. Online transaction processing (OLTP) systems typically use very little PGA memory compared to data warehouse (DW) or decision support systems (DSS). Using the existing memory efficiently also includes tuning the applications. A poorly tuned application can use large quantities of memory. For example, an application that uses frequent full table scans because indexes do not exist or are unusable can cause a large amount of I/O, reducing performance. The first and most effective tuning technique is to tune high cost SQL statements.

Enterprise Manager Cloud Control and Enterprise Manager Database Express both provide Memory Advisors. These tools monitor the memory usage by the SGA components and PGA and project the differences in terms of efficiency for increased and decreased memory allocations. These projections use the current workload. They can help you size the SGA based on the activity in your particular database. The advisors make sizing recommendations for manual settings, when the automatic memory management is disabled. These same advisors provide input to automatic memory management to determine the most efficient component sizes.

Automatic Memory Management

With Automatic Memory Management, the database server can size the SGA and PGA automatically according to your workload.



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Automatic Memory Management (AMM) allows the Oracle Database server to manage SGA memory and instance PGA memory sizing automatically. To do so (on most platforms), you set only a target memory size initialization parameter (`MEMORY_TARGET`) and a maximum memory size initialization parameter (`MEMORY_MAX_TARGET`), and the database server dynamically exchanges memory between the SGA and the instance PGA, as needed, to meet processing demands.

With this memory management method, the database server also dynamically tunes the sizes of the individual SGA components and the sizes of the individual PGAs.

Because the target memory initialization parameter is dynamic, you can change the target memory size at any time without restarting the database instance. The maximum memory size serves as an upper limit so that you cannot accidentally set the target memory size too high. Because certain SGA components either cannot easily shrink or must remain at a minimum size, the database server also prevents you from setting the target memory size too low.

This indirect memory transfer relies on the operating system (OS) mechanism of freeing shared memory. After memory is released to the OS, the other components can allocate memory by requesting memory from the OS. Currently, Automatic Memory Management is implemented on Linux, Solaris, HPUX, AIX, and Windows.

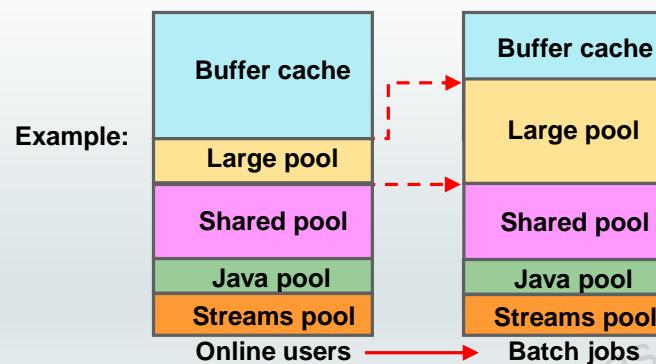
Oracle recommends the use of AMM unless you have special requirements.

Use the following views to monitor Automatic Memory Management:

- **V\$MEMORY_DYNAMIC_COMPONENTS:** Shows the current sizes of all dynamically tuned memory components, including the total sizes of the SGA and instance PGA
- **V\$MEMORY_RESIZE_OPS:** Shows a circular history buffer of the last 800 memory resize requests
- **V\$MEMORY_TARGET_ADVICE:** Provides tuning advice for the `MEMORY_TARGET` initialization parameter

Automatic Shared Memory Management

- Automatically adapts to workload changes
- Maximizes memory utilization
- Helps eliminate out-of-memory errors



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

If you need a fixed PGA, consider the use of Automatic Shared Memory Management (ASMM), which also simplifies SGA memory management. You specify the total amount of SGA memory available to an instance by using the `SGA_TARGET` initialization parameter, and the Oracle Database server automatically distributes this memory among the various SGA components to ensure the most effective memory utilization.

For example, in a system that runs large online transactional processing (OLTP) jobs during the day (requiring a large buffer cache) and runs parallel batch jobs at night (requiring a large value for the large pool), you would have to simultaneously configure both the buffer cache and the large pool to accommodate your peak requirements.

With ASMM, when the OLTP job runs, the buffer cache uses most of the memory to allow for good I/O performance. When the data analysis and reporting batch job starts up later, the memory is automatically migrated to the large pool so that it can be used by parallel query operations without producing memory overflow errors.

The Oracle Database server remembers the sizes of the automatically tuned components across instance shutdowns if you are using a server parameter file (SPFILE). As a result, the system needs to learn the characteristics of the workload again each time an instance is started. It can begin with information from the past instance and continue evaluating the workload where it left off at the last shutdown.

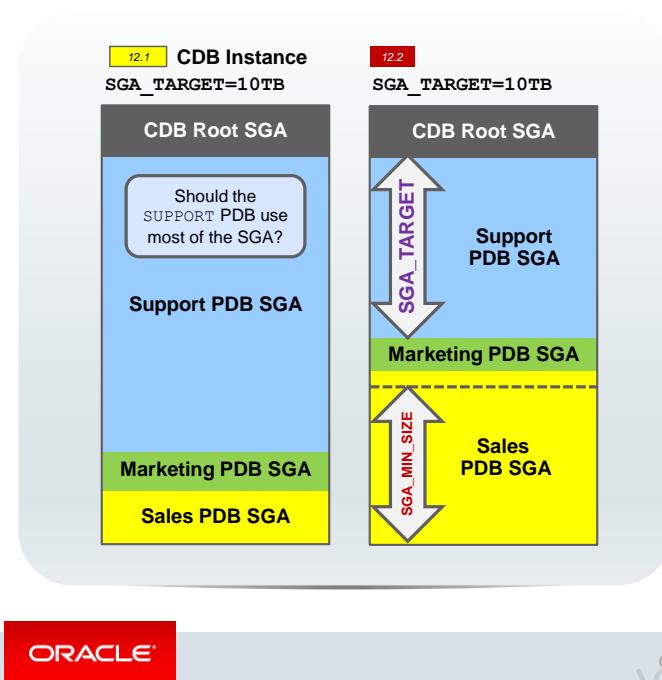
The Automatic Shared Memory Management feature uses the SGA memory broker that is implemented by two background processes: Manageability Monitor (MMON) and Memory Manager (MMAN). Statistics and memory advisory data are periodically captured in memory by MMON. MMAN coordinates the sizing of the memory components according to MMON decisions. The SGA memory broker keeps track of the sizes of the components and pending resize operations.

The SGA memory broker observes the system and workload in order to determine the ideal distribution of memory. It performs this check every few minutes so that memory can always be present where needed. In the absence of Automatic Shared Memory Management, components had to be sized to anticipate their individual worst-case memory requirements.

On the basis of workload information, Automatic Shared Memory Management:

- Captures statistics periodically in the background
- Uses memory advisors
- Performs what-if analysis to determine the best distribution of memory
- Moves memory to where it is most needed
- Saves component sizes after shutdown if an SPFILE is used (the sizes can be resurrected from before the last shutdown)

Managing the SGA for PDBs



- **SGA_TARGET** set at PDB level enforces a hard limit for the PDB's SGA.
- **SGA_TARGET** at PDB level provides more SGA for other containers.
- **SGA_MIN_SIZE** set for a PDB guarantees SGA space for the PDB.
- Parameters at PDB level:
 - **DB_CACHE_SIZE**
 - **SHARED_POOL_SIZE**
- PDB minimums cannot be > 50% of memory

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

In a CDB, there is one SGA allocation for the instance, shared by all containers including the CDB root and all PDBs. Most of the SGA is a cache that favors frequently accessed objects in the buffer cache, the shared pool, and the in-memory column store. In Oracle Database 12c Release 1, active PDBs dominate the space in the SGA cache. In the graphic, the Support PDB has an active memory-intensive workload. It monopolizes the SGA. The Marketing PDB needs very little SGA. Sales PDB performance depends on critical buffer cache data and parsed cursors. The Support PDB is more active and is evicting its data.

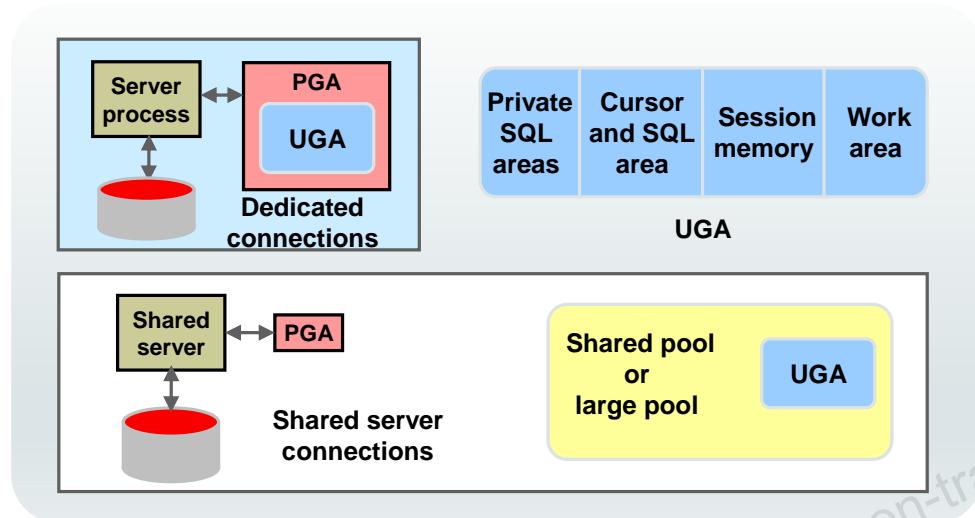
Oracle Database 12c Release 2 provides SGA and PGA memory management at the PDB level:

- Setting an **SGA_TARGET** for a PDB enforces a hard limit for the PDB's SGA and provides more SGA for the other containers within the CDB. The sum of all PDB **SGA_TARGET** parameter values does not necessarily need to be less than the instance **SGA_TARGET** value, but each PDB **SGA_TARGET** value cannot exceed the instance **SGA_TARGET**, nor **SGA_MAX_SIZE**. **SGA_TARGET** for PDBs works only if the CDB's **SGA_TARGET** parameter is set.
- Setting **DB_CACHE_SIZE** and **SHARED_POOL_SIZE** guarantees minimum sizes for the PDB.
- Setting an **SGA_MIN_SIZE** for a PDB guarantees the SGA space for the PDB.

SGA_TARGET, **DB_CACHE_SIZE**, and **SHARED_POOL_SIZE** do not work if **MEMORY_TARGET** is set for the CDB.

No more than 50% of the memory can be set aside for the PDB minimums: **SGA_MIN_SIZE**, **DB_CACHE_SIZE**, and **SHARED_POOL_SIZE**.

Managing the Program Global Area (PGA)



Automatic PGA memory management is enabled by default.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The program global area (PGA) is a memory region that contains data and control information for a server process. It is nonshared memory created by the Oracle server when a server process is started. Access to it is exclusive to that server process. The total PGA memory allocated by all server processes attached to an Oracle instance is also referred to as the aggregated PGA memory allocated by the instance.

Part of the PGA can be located in the SGA when using shared servers.

PGA memory typically contains the following:

- **Private SQL Area:** A private SQL area, also called the user global area (UGA), contains data, such as bind information and runtime memory structures. This information is specific to each session's invocation of the SQL statement; bind variables hold different values, and the state of the cursor is different, among other things. Each session that issues a SQL statement has a private SQL area. Each user that submits the same SQL statement has his or her own private SQL area that uses a single shared SQL area. Thus, many private SQL areas can be associated with the same shared SQL area. The location of a private SQL area depends on the type of connection established for a session. If a session is connected through a dedicated server, private SQL areas are located in the server process's PGA. However, if a session is connected through a shared server, part of the private SQL area is kept in the SGA.
- **Cursor and SQL Areas:** The application developer of an Oracle Pro*C program or Oracle Call Interface (OCI) program can explicitly open cursors or handles to specific private SQL areas and use them as a named resource throughout the execution of the program. Recursive cursors that the database issues implicitly for some SQL statements also use shared SQL areas.

- **Work Area:** For complex queries (for example, decision support queries), a big portion of the PGA is dedicated to work areas allocated by memory-intensive operators. A sort operator uses a work area (the sort area) to perform the in-memory sort of a set of rows. Similarly, a hash-join operator uses a work area (the hash area) to build a hash table from its left input. The size of a work area can be controlled and tuned. Generally, bigger work areas can significantly improve the performance of a particular operator at the cost of higher memory consumption.
- **Session Memory:** Session memory is the memory allocated to hold a session's variables (logon information) and other information related to the session. For a shared server, the session memory is shared and not private.

By default, Oracle Database automatically and globally manages the total amount of memory dedicated to the instance PGA. You can control this amount by setting the `PGA_AGGREGATE_TARGET` initialization parameter. Oracle Database then tries to ensure that the total amount of PGA memory allocated across all database server processes and background processes never exceeds this target. But this is a target value and not a hard limit. `PGA_AGGREGATE_LIMIT` sets a hard limit for the amount of PGA that can be used. The minimum value is 1024 MB and the maximum is 120% of physical memory minus the total SGA, and it must be at least as large as `PGA_AGGREGATE_TARGET`. If `PGA_AGGREGATE_LIMIT` is not set, it defaults to 200% of `PGA_AGGREGATE_TARGET` within the same minimum and maximum as stated. When `PGA_AGGREGATE_LIMIT` is exceeded, the sessions using the most memory will have their calls aborted. Parallel queries will be treated as a unit. If the total PGA memory usage is still over the limit, sessions using the most memory will be terminated. SYS processes and fatal background processes are exempt from this limit.

Managing the PGA for PDBs

Instance PGA_AGGREGATE_LIMIT

- No more PGA can be allocated.
- Calls or sessions of the largest PGA users are terminated.

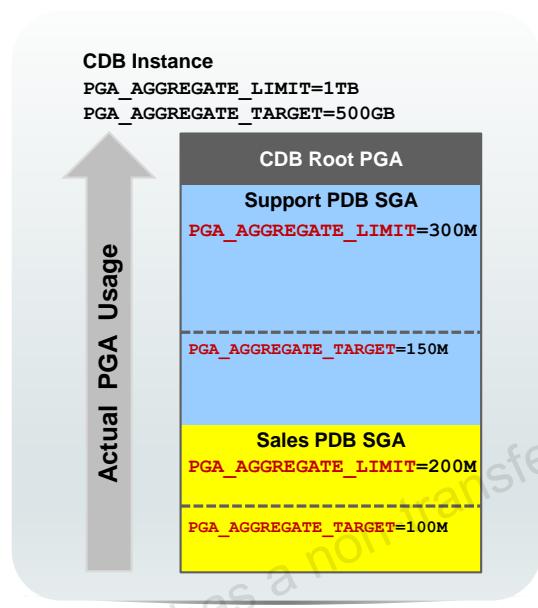
Instance PGA_AGGREGATE_TARGET

- All sessions must use TEMP rather than PGA.

PDB PGA_AGGREGATE_LIMIT

PDB PGA_AGGREGATE_TARGET

- These parameters set the same behavior at the PDB level.



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

You can set the following the parameters at the PDB level:

- PGA_AGGREGATE_TARGET: Specifies the target aggregate PGA memory available to all server processes attached to the instance
- PGA_AGGREGATE_LIMIT: Specifies a hard limit for aggregate PGA memory

Summary

In this lesson, you should have learned how to:

- Describe the activities that you perform to manage database performance
- Use performance views and tools to monitor database instance performance
- Describe the Oracle performance tuning methodology
- Describe statistics and metrics that are collected by the Oracle Database server
- Configure and monitor memory components for optimal performance



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 20: Overview

- 20-1: Managing Performance
- 20-2: Resolving Lock Conflicts



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2020, Oracle and/or its affiliates.

Tarushi Arora (tarushi.arora@oracle.com) has a non-transferable
license to use this Student Guide.

Tuning SQL

The ORACLE logo, consisting of the word "ORACLE" in white capital letters on a red rectangular background.

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

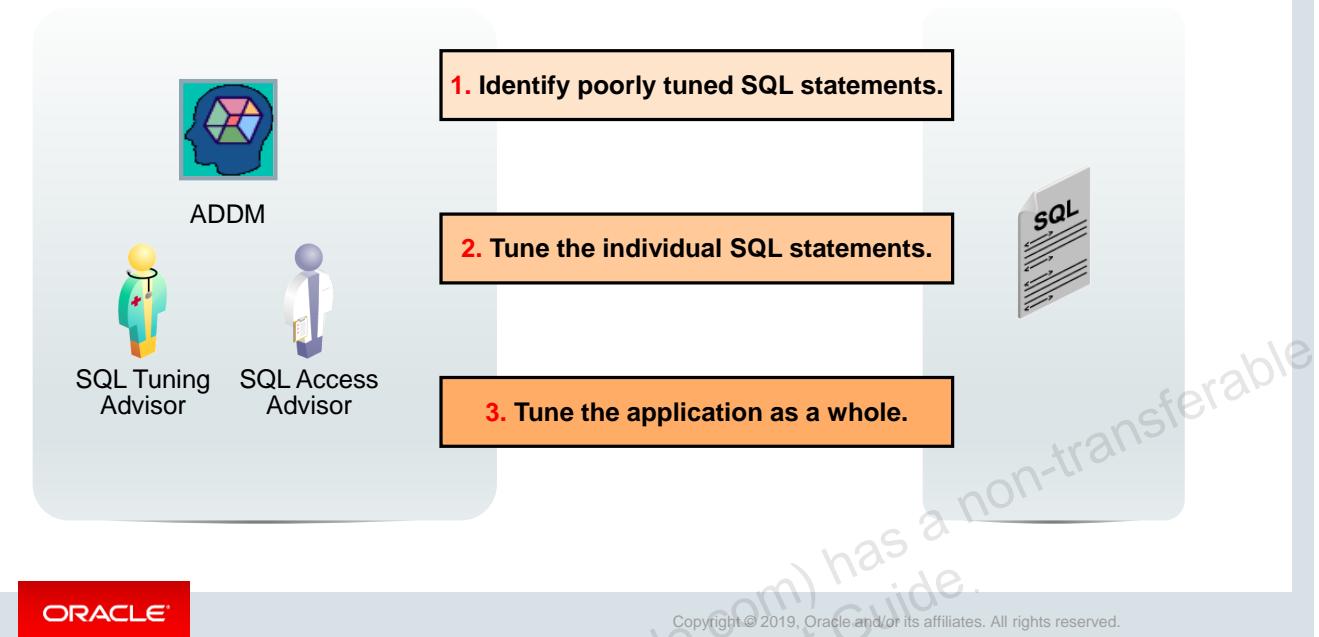
- Describe the SQL tuning methodology
- Manage optimizer statistics
- Use SQL Tuning Advisor to identify and tune SQL statements that are using the most resources
- Use SQL Access Advisor to tune a workload



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

SQL Tuning Process

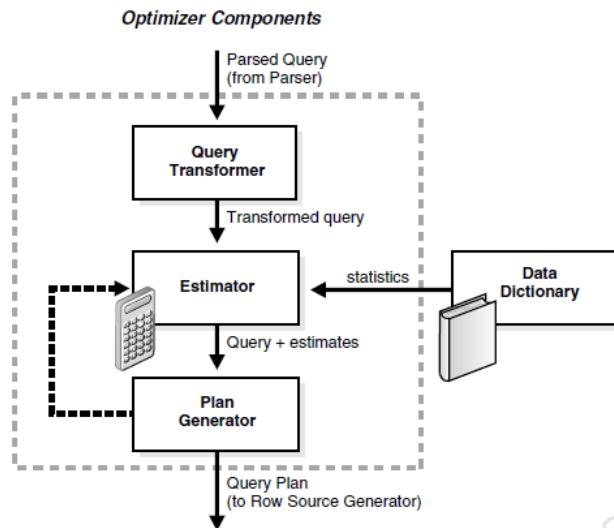


SQL Tuning Process

1. **Identify poorly tuned SQL statements:** Generally, the tuning effort that yields the most benefit is SQL tuning. Poorly tuned SQL uses more resources than required. This inefficiency prevents scalability, uses more OS and database resources, and increases response time. To tune poorly tuned SQL statements, they must be identified and then tuned. SQL statements can be tuned individually, but often the solution that optimizes one statement can hurt the performance of several others. The SQL statements that use the most resources are, by definition, the statements in need of tuning. These are statements that have the longest elapsed time, use the most CPU, or do the most physical or logical reads. Automatic Database Diagnostic Monitor (ADDM) can detect high-load SQL statements.
2. **Tune the individual statements:** Tune the individual statements by checking the optimizer statistics, check the explain plan for the most efficient access path, test alternative SQL constructions, and test possible new indexes, materialized views, and partitioning. SQL Tuning Advisor and SQL Access Advisor, described later in this lesson, can help with this task.
3. **Tune the application as a whole:** Test the application as a whole by using the tuned SQL statements. Is the overall performance better?

The methodology is sound, but tedious. Tuning an individual statement is not difficult. Testing the overall impact of the individual statement tuning on an application can be very difficult.

Oracle Optimizer



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The optimizer is the part of the Oracle Database server that creates the execution plan for a SQL statement. The determination of the execution plan is an important step in the processing of any SQL statement and can greatly affect execution time.

The execution plan is a series of operations that are performed in sequence to execute the statement. The optimizer considers many factors related to the referenced objects and the conditions specified in the query. The information necessary to the optimizer includes:

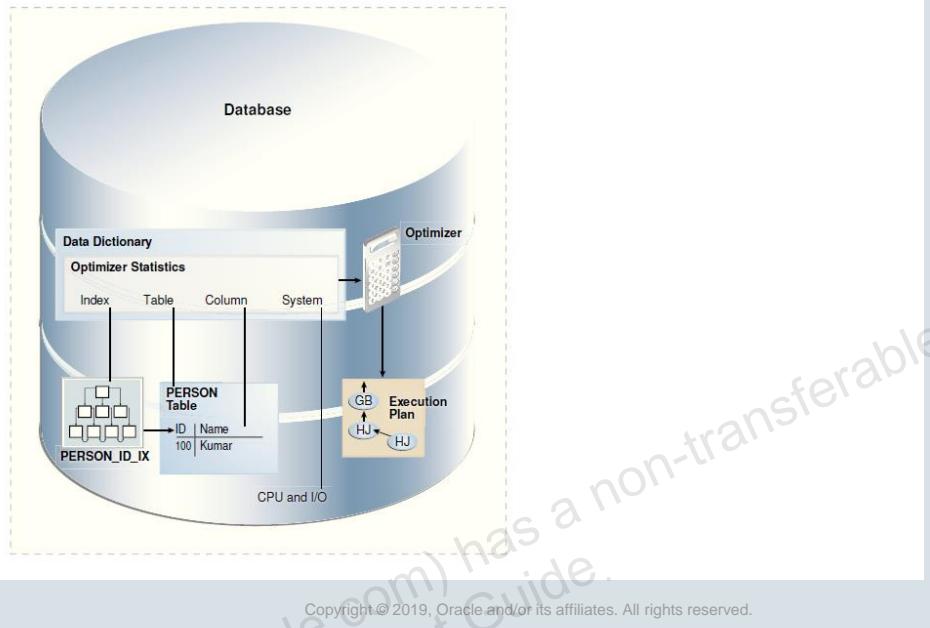
- Statistics gathered for the system (I/O, CPU, and so on) as well as schema objects (number of rows, index, and so on)
- Information in the dictionary
- WHERE clause qualifiers
- Hints supplied by the developer

The optimizer:

- Evaluates expressions and conditions
- Uses object and system statistics
- Decides how to access the data and join tables
- Determines the most efficient path

When you use diagnostic tools, such as Enterprise Manager, EXPLAIN PLAN, and SQL*Plus AUTOTRACE, you can see the execution plan that the optimizer chooses.

Optimizer Statistics



Optimizer statistics include table, column, index, and system statistics. Statistics for tables and indexes are stored in the data dictionary. These statistics are not intended to provide real-time data. They provide the optimizer a statistically correct snapshot of data storage and distribution, which the optimizer uses to make decisions on how to access data.

The statistics that are collected include:

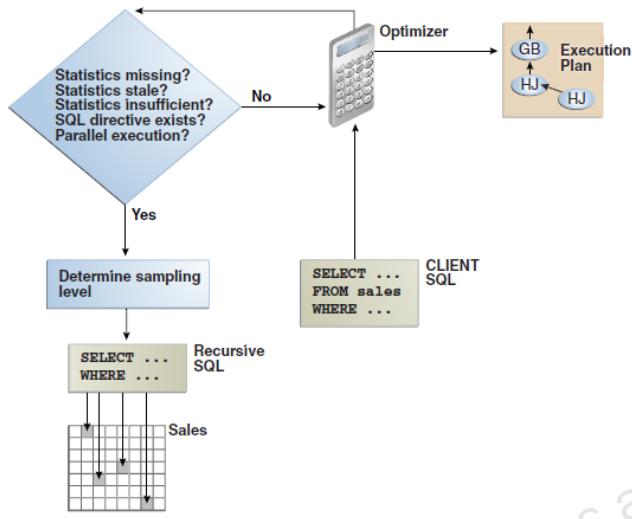
- Size of the table or index in database blocks
- Number of rows
- Average row size and chain count (tables only)
- Height and number of deleted leaf rows (indexes only)

As data is inserted, deleted, and modified, these statistics change. Because the performance impact of maintaining real-time data distribution statistics is prohibitive, these statistics are updated by periodically gathering statistics on tables and indexes.

Optimizer statistics are collected automatically by an automatic maintenance job that runs during predefined maintenance windows once daily by default. System statistics are operating system characteristics that are used by the optimizer. These statistics are not collected automatically. For details about collecting system statistics, see the *Oracle Database Performance Tuning Guide*.

Optimizer statistics are not the same as the database performance statistics that are gathered in the Automatic Workload Repository (AWR) snapshot.

Optimizer Statistics Collection



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Optimizer statistics are collections of data that are specific details about database objects. These statistics are essential for the query optimizer to choose the best execution plan for each SQL statement. These statistics are gathered periodically and do not change between gatherings.

Statistics can be collected in the following ways:

- Automatically: Automatic Maintenance Tasks
- Manually: DBMS_STATS package
- By setting database initialization parameters
- By importing statistics from another database

The recommended approach to gathering optimizer statistics is to allow the Oracle Database server to automatically gather the statistics. Automatic Maintenance Tasks can be created automatically at database creation time and is managed by the Scheduler. It gathers statistics on all objects in the database that have either missing or stale optimizer statistics by default. You can change the default configuration through the Automatic Maintenance Tasks page.

System statistics describe the system's hardware characteristics, such as I/O and CPU performance and utilization, to the query optimizer. When choosing an execution plan, the optimizer estimates the I/O and CPU resources required for each query. System statistics enable the query optimizer to more accurately estimate I/O and CPU costs and thereby choose a better execution plan. System statistics are collected by using the DBMS_STATS.GATHER_SYSTEM_STATS procedure. When the Oracle Database server gathers system statistics, it analyzes system activity in a specified period of time. System statistics are not automatically gathered. Oracle Corporation recommends that you use the DBMS_STATS package to gather system statistics.

If you choose not to use automatic statistics gathering, you must manually collect statistics in all schemas, including system schemas. If the data in your database changes regularly, you also need to gather statistics regularly to ensure that the statistics accurately represent characteristics of your database objects. To manually collect statistics, use the DBMS_STATS package. This PL/SQL package is also used to modify, view, export, import, and delete statistics.

You can also manage optimizer and system statistics collection through database initialization parameters. For example:

- The OPTIMIZER_DYNAMIC_SAMPLING parameter controls the level of dynamic sampling performed by the optimizer. You can use dynamic sampling to estimate statistics for tables and relevant indexes when they are not available or are too out of date to trust. Dynamic sampling also estimates single-table predicate selectivity when collected statistics cannot be used or are likely to lead to significant errors in estimation.
- The STATISTICS_LEVEL parameter controls all major statistics collections or advisories in the database and sets the statistics collection level for the database. The values for this parameter are BASIC, TYPICAL, and ALL. You can query the V\$STATISTICS_LEVEL view to determine which parameters are affected by the STATISTICS_LEVEL parameter.

Note: Setting STATISTICS_LEVEL to BASIC disables many automatic features and is not recommended.

Setting Optimizer Statistics Preferences

DBMS_STATS.GATHER_*_STATS procedures: Gather statistics for an entire database or for individual objects using default values

Use the SET_*_PREFS procedures to create preference values for any object that is not owned by SYS or SYSTEM

Query DBA_TAB_STAT_PREFS to view object-level preferences

Execute the DBMS_STATS.GET_PREFS procedure for each preference to see the global preferences



Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The DBMS_STATS.GATHER_*_STATS procedures can be called at various levels to gather statistics for an entire database or for individual objects, such as tables. When the GATHER_*_STATS procedures are called, several of the parameters are often allowed to default. The supplied defaults work well for most of the objects in the database, but for some objects or schemas, the defaults need to be changed. Instead of running manual jobs for each of these objects, Oracle Database enables you to set values (called preferences) for individual objects, schemas, or databases or to change the default values with a global-level command.

The preferences specify the parameters that are given to the gather procedures. The SET_*_PREFS procedures create preference values for any object that is not owned by SYS or SYSTEM. The expected use is that the DBA will set the global preferences for any parameters that should be database-wide. These will be applied for any parameter that is allowed to default.

The SET_DATABASE_PREFS procedure iterates over all the tables and schemas in the database, setting the specified preference. SET_SCHEMA_PREFS iterates over the tables in the specified schema. SET_TABLE_PREFS sets the preference value for a single table.

All object preferences—whether set at the database, schema, or table level—are held in a single table. Changing the preferences at the schema level overwrites the preferences that were previously set at the table level.

When the various gather procedures execute, they retrieve the object-level preferences that were set for each object. You can view the object-level preferences in the DBA_TAB_STAT_PREFS view. Any preferences that are not set at the object level will be set to the global-level preferences. You can see the global preferences by calling the DBMS_STATS.GET_PREFS procedure for each preference.

For details about these preferences, see the `DBMS_STATS` documentation in the *Oracle Database PL/SQL Packages and Types Reference*.

Preferences may be deleted with the `DBMS_STATS.DELETE_*_PREFS` procedures at the table, schema, and database levels. You can reset the global preferences to the recommended values with the `DBMS_STATS.RESET_PARAM_DEFAULTS` procedure.

Optimizer Statistics Advisor

- If best practices change in a new release, Optimizer Statistics Advisor encodes these practices in its **rules**.
- The advisor always provides the most up-to-date recommendations.
- Track and analyze how statistics are collected.
 - Class of findings: System, Operations, Objects
- Scope of findings
 - Problems with gathering of statistics
 - Status of automatic statistic gathering jobs
 - Quality of current statistics
- Suggestion for changes to the statistics collection



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

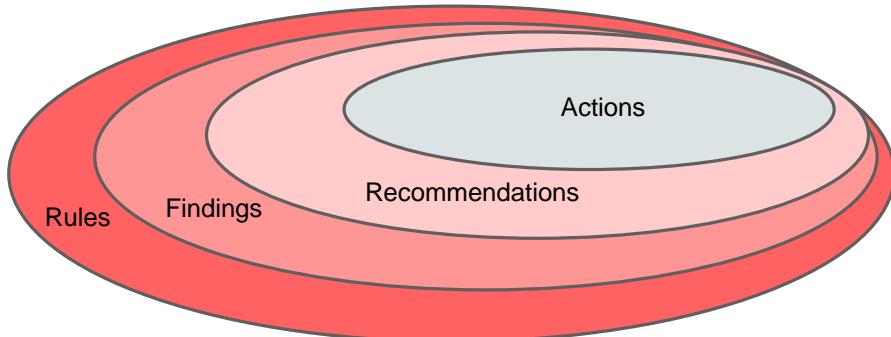
Since the introduction of the cost-based optimizer, optimizer statistics play a significant part in determining the execution plan for queries. Therefore, it is critical for the optimizer to have accurate and up-to-date statistics. The DBMS_STATS package serves this purpose and is improved in every release by adding new features. However, under many circumstances, these new features have not been fully utilized by customers or are being used in incorrect ways. Customers often use scripts and settings from one release to the next, based on earlier experience. These settings and methods may have been superseded or produce statistics that no longer give the most effective optimizer results. Optimizer Statistics Advisor uses rules consistent with the current release to recommend changes to the way statistics are being gathered.

The advisor has a set of rules or recommended practices that are compared against the current statistics to generate findings. The rules are applied at the system, operation, or object level, such as whether the Automatic Gather Statistics jobs are scheduled, the statistics gathering procedures are using default parameters, and statistics are consistent across related objects. These rules check on issues related to the gathering of statistics—the schedules, parameters, and errors related to the automatic statistics gathering jobs. The rules include a variety of object-related issues, including whether incremental mode setting is efficient.

Optimizer Statistics Advisor Report

Report sections:

- Header
- Summary
- Errors
- Findings



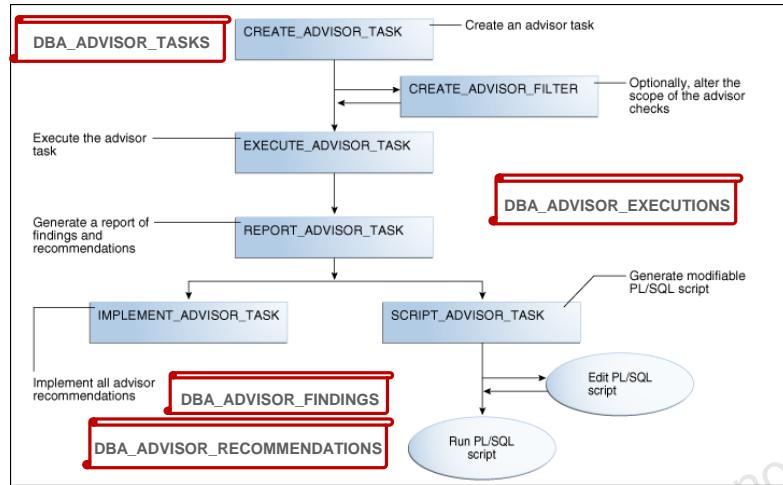
Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Optimizer Statistics Advisor report has four basic concepts:

- **Rules:** Check the current configuration, history, and current statistics. Rules are added and changed by release to reflect best practices.
- **Findings:** They are generated when rules are not followed. An individual rule may generate many findings, but each finding is generated by only one rule. Some findings may be informational only, such as object staleness.
- **Recommendations:** They are responses the customer could make to resolve the finding. It is possible that several recommendations could be generated, and further investigation would be required by the customer. One or more rationales are given for each recommendation. Not all findings generate recommendations.
- **Actions:** They are PL/SQL statements or commands that the user can simply run in the command line to solve problems. They are provided in the form of scripts. Not all recommendations generate actions. For some recommendations, it is not possible to generate an action.

The report has sections for header, summary, errors, and findings. The header includes the advisor task parameters. The summary lists findings, and the errors section lists any errors the task encountered. The findings section includes the rule, findings, recommendations, and actions for each rule that produces a finding.

Executing Optimizer Statistics Advisor Tasks



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

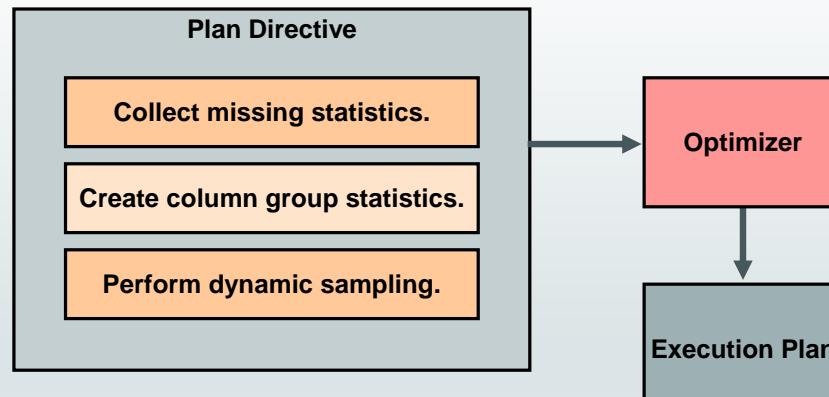
An Optimizer Statistics Advisor task can be executed with PL/SQL calls. Each task must be provided a unique task name. The definition of the `CREATE_ADVISOR_TASK()` function parameters are:

- **TASK_NAME**: Name of the Statistics Advisor task
- **TIME_LIMIT**: The maximum duration the task can run

A filter list can be applied to the task to limit the scope of an advisor task using inclusion or exclusion lists for a user-specified set of rules, schemas, or operations. System rules are always checked. For example, you can configure an advisor task to include only recommendations for the SH schema. Also, you could exclude all violations of the rule for stale statistics.

You can create filters with the following `DBMS_STATS` procedures either individually or in combination: `CONFIGURE_ADVISOR_OBJ_FILTER`, `CONFIGURE_ADVISOR_RULE_FILTER`, and `CONFIGURE_ADVISOR_OPR_FILTER`. An Optimizer Statistics Advisor report can be generated with PL/SQL calls. The `REPORT_ADVISOR_TASK` function produces a report in text, HTML, or XML format. The `IMPLEMENT_ADVISOR_TASK` implements the recommendations of the task based on the filters in place. An additional parameter, `LEVEL`, can be set to either `TYPICAL` or `ALL`. `TYPICAL` is the default. `ALL` ignores the filters.

SQL Plan Directives



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

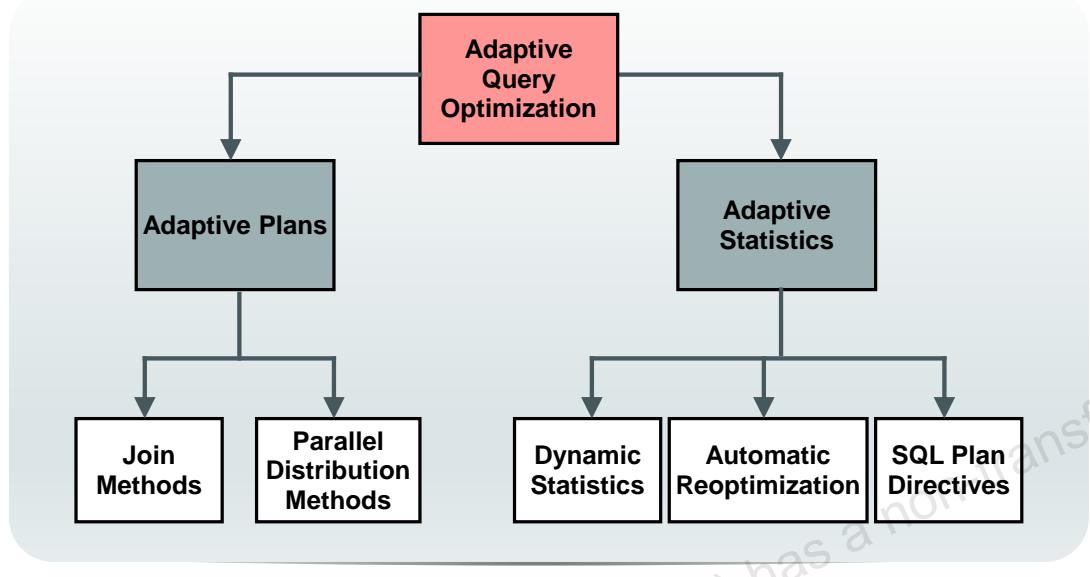
The Oracle Database 12c server can use a SQL plan directive, which is additional information and instructions that the optimizer can use to generate a more optimal plan. For example, a SQL plan directive might instruct the optimizer to collect missing statistics, create column group statistics, or perform dynamic sampling. During SQL compilation or execution, the database analyzes the query expressions that are missing statistics or that misestimate optimizer cardinality to create SQL plan directives. When the optimizer generates an execution plan, the directives give the optimizer additional information about objects that are referenced in the plan.

SQL plan directives are not tied to a specific SQL statement or SQL ID. The optimizer can use SQL plan directives for SQL statements that are nearly identical because SQL plan directives are defined on a query expression. For example, directives can help the optimizer with queries that use similar patterns, such as web-based queries that are the same except for a select list item. The database stores SQL plan directives persistently in the SYSAUX tablespace. When generating an execution plan, the optimizer can use SQL plan directives to obtain more information about the objects that are accessed in the plan.

Directives are automatically maintained, created as needed, and purged if not used after a year.

Directives can be monitored in `DBA_SQL_PLAN_DIR_OBJECTS`. SQL plan directives improve plan accuracy by persisting both compilation and execution statistics in the SYSAUX tablespace, allowing them to be used by multiple SQL statements.

Adaptive Execution Plans



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

The Adaptive Execution Plans feature enables the optimizer to automatically adapt a poorly performing execution plan at run time and prevent a poor plan from being chosen on subsequent executions. The optimizer instruments its chosen plan so that at run time, it can be detected if the optimizer's estimates are not optimal. Then the plan can be automatically adapted to the actual conditions. An adaptive plan is a plan that changes after optimization when optimizer estimates prove inaccurate.

The optimizer can adapt plans based on statistics that are collected during statement execution. All adaptive mechanisms can execute a plan that differs from the plan that was originally determined during hard parse. This improves the ability of the query-processing engine (compilation and execution) to generate better execution plans.

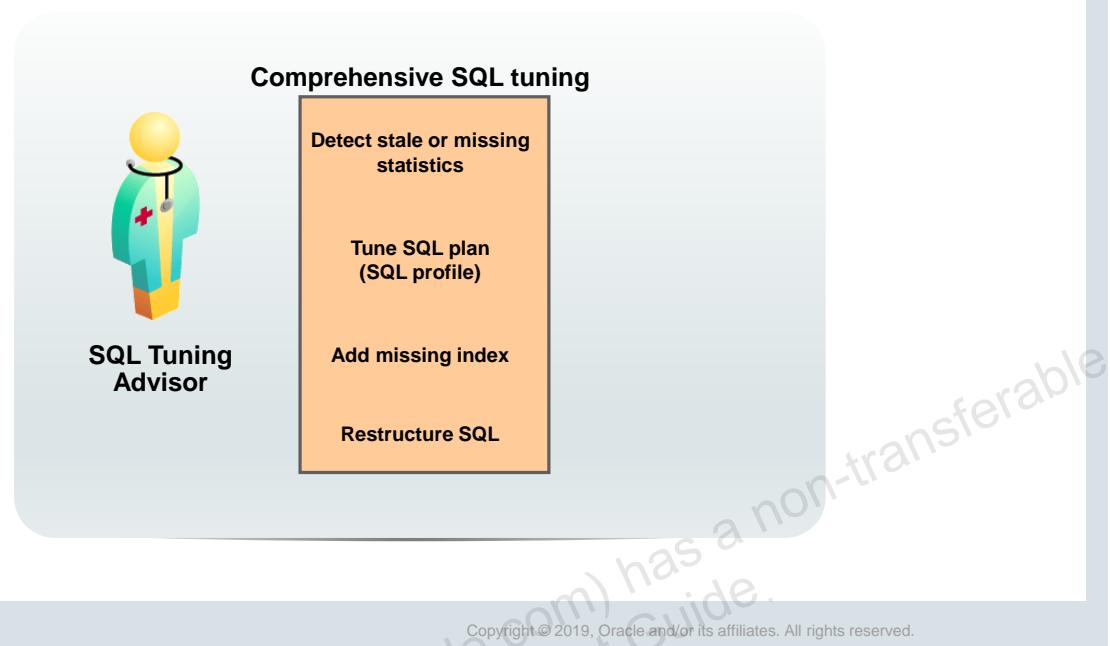
The two Adaptive Execution Plan techniques are:

- **Dynamic plans:** A dynamic plan chooses among subplans during statement execution. For dynamic plans, the optimizer must decide which subplans to include in a dynamic plan, which statistics to collect to choose a subplan, and thresholds for this choice.
- **Re-optimization:** In contrast, re-optimization changes a plan for executions after the current execution. For re-optimization, the optimizer must decide which statistics to collect at which points in a plan and when re-optimization is feasible.

Note: OPTIMIZER_ADAPTIVE_REPORTING_ONLY controls reporting-only mode for adaptive optimizations. When set to TRUE, adaptive optimizations run in reporting-only mode where the information required for an adaptive optimization is gathered, but no action is taken to change the plan.

The optimizer can pick the best-performing plan during any execution of the statement, not just the first execution. If the underlying data changes, or if queries re-execute with different input data, then the optimizer can adapt its plan to match the statistics gathered in the current execution. The continuous adaptive query plan adapts for every execution of the same cursor instead of only once.

SQL Tuning Advisor: Overview



SQL Tuning Advisor is the primary driver of the tuning process. It performs several types of analyses:

- **Statistics Analysis:** Checks each query object for missing or stale statistics and makes recommendations to gather relevant statistics
- **SQL Profiling:** The optimizer verifies its own estimates and collects auxiliary information to remove estimation errors. It builds a SQL profile by using the auxiliary information and makes a recommendation to create it. When a SQL profile is created, it enables the query optimizer to generate a well-tuned plan.
- **Access Path Analysis:** New indexes are considered if they significantly improve access to each table in the query. When appropriate, recommendations to create such objects are made.
- **SQL Structure Analysis:** SQL statements that use bad plans are identified and relevant suggestions are made to restructure them. The suggested changes can be syntactic as well as semantic.

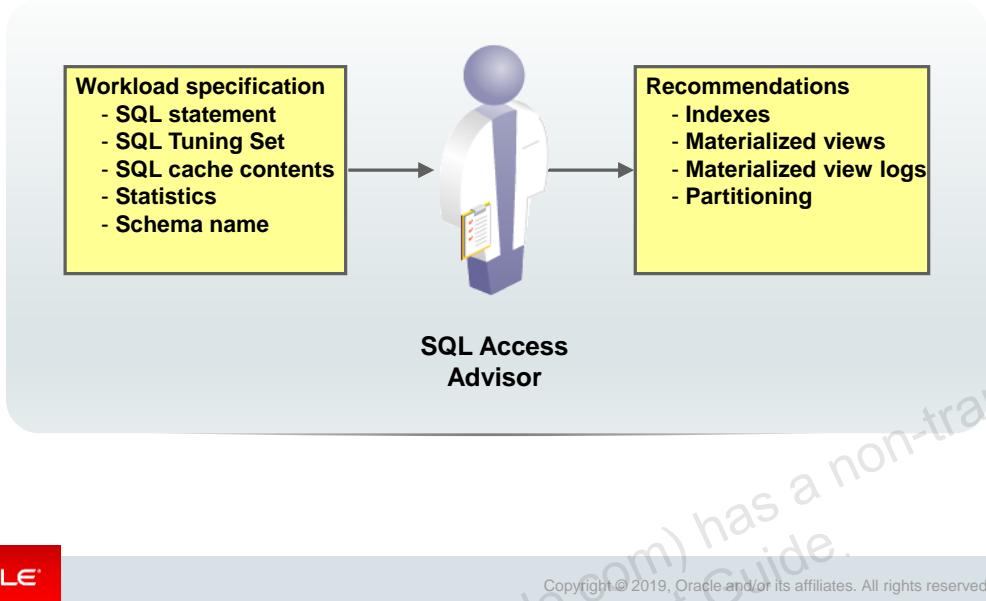
SQL Tuning Advisor considers each SQL statement included in the advisor task independently. Creating a new index may help a query, but may hurt the response time of DML. So, a recommended index or other object should be checked with SQL Access Advisor over a workload (a set of SQL statements) to determine whether there is a net gain in performance.

SQL Tuning Advisor runs automatically every night as the Automatic SQL Tuning Task. There may be times when a SQL statement needs immediate tuning action. You can use SQL Tuning Advisor to analyze SQL statements and obtain performance recommendations at any time. Typically, you run this advisor as an ADDM performance-finding action.

Additionally, you can run SQL Tuning Advisor when you want to analyze the top SQL statements consuming the most CPU time, I/O, and memory.

Even though you can submit multiple statements to be analyzed in a single task, each statement is analyzed independently. To obtain tuning recommendations that consider the overall performance of a set of SQL, use SQL Access Advisor.

SQL Access Advisor: Overview



ORACLE

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

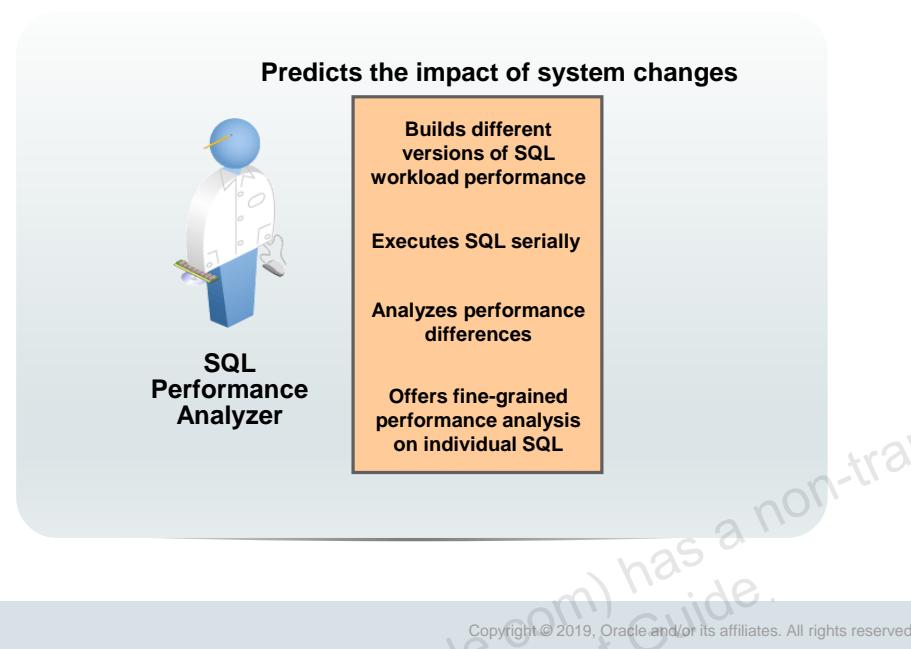
SQL Access Advisor can recommend the proper set of materialized views, materialized view logs, partitioning, and indexes for a given workload. Understanding and using these structures is essential when optimizing SQL because they can result in significant performance improvements in data retrieval.

SQL Access Advisor recommends bitmap, function-based, and B-tree indexes. A bitmap index offers a reduced response time for many types of ad hoc queries and reduced storage requirements compared to other indexing techniques. B-tree indexes are most commonly used in a data warehouse to index unique or near-unique keys.

Another component of SQL Access Advisor also recommends how to optimize materialized views so that they can be fast refreshable and take advantage of general query rewrite.

Note: For more information about materialized views and query rewrite, see *Oracle Database Performance Tuning Guide*.

SQL Performance Analyzer: Overview



Oracle Database includes SQL Performance Analyzer, which gives you an exact and accurate assessment of the impact of change on the SQL statements that make up the workload. SQL Performance Analyzer helps you forecast the impact of a potential change on the performance of a SQL query workload. This capability provides you with detailed information about the performance of SQL statements, such as before-and-after execution statistics, and statements with performance improvement or degradation. This enables you (for example) to make changes in a test environment to determine whether the workload performance will be improved through a database upgrade.

SQL Performance Analyzer includes the following capabilities:

- Helps predict the impact of system changes on SQL workload response time
- Builds different versions of SQL workload performance (that is, SQL execution plans and execution statistics)
- Executes SQL serially (concurrency not honored)
- Analyzes performance differences
- Offers fine-grained performance analysis on individual SQL
- Is integrated with SQL Tuning Advisor to tune regressions

Use Cases

SQL Performance Analyzer can be used to predict and prevent potential performance problems for any database environment change that affects the structure of the SQL execution plans. The changes can include (but are not limited to) any of the following:

- Database upgrades
- Implementation of tuning recommendations
- Schema changes
- Statistics gathering
- Database parameter changes
- OS and hardware changes

You can use SQL Performance Analyzer to predict SQL performance changes that result from changes for even the most complex environments. As applications evolve through the development life cycle, database application developers can test changes to schemas, database objects, and rewritten applications to mitigate any potential performance impact.

SQL Performance Analyzer also enables the comparison of SQL performance statistics.

You can access SQL Performance Analyzer through Enterprise Manager or by using the DBMS_SQLPA package.

For details about the DBMS_SQLPA package, see the *Oracle Database PL/SQL Packages and Types Reference Guide*.

Summary

In this lesson, you should have learned how to:

- Describe the SQL tuning methodology
- Manage optimizer statistics
- Use SQL Tuning Advisor to identify and tune SQL statements that are using the most resources
- Use SQL Access Advisor to tune a workload



ORACLE®

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Practice 21: Overview

- 21-1: Using SQL Tuning Advisor
- 21-2: Using Optimizer Statistics Advisor



Copyright© 2019, Oracle and/or its affiliates. All rights reserved.